

***IN THE UNITED STATES PATENT AND TRADEMARK OFFICE***

Applicant: Atsufumi SHIBAYAMA et al.  
Title: PROGRAM PARALLELIZATION DEVICE, PROGRAM  
PARALLELIZATION METHOD, AND PROGRAM  
PARALLELIZATION PROGRAM  
Appl. No.: Unassigned  
Filing Date: March 30, 2004  
Examiner: Unassigned  
Art Unit: Unassigned

**CLAIM FOR CONVENTION PRIORITY**

Commissioner for Patents  
PO Box 1450  
Alexandria, Virginia 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign application:

Japanese Patent Application No. 2003-093076  
filed 03/31/2003.

Respectfully submitted,

Date: March 30, 2004

FOLEY & LARDNER LLP  
Customer Number: 22428  
Telephone: (202) 672-5407  
Facsimile: (202) 672-5399

By Phillip J. Anticola *Reg. No.*  
38,819  
for / David A. Blumenthal  
Attorney for Applicant  
Registration No. 26,257

日 本 国 特 許 庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日            2 0 0 3 年   3 月 3 1 日  
Date of Application:

出 願 番 号            特 願 2 0 0 3 - 0 9 3 0 7 6  
Application Number:  
[ST. 10/C] :            [ J P 2 0 0 3 - 0 9 3 0 7 6 ]

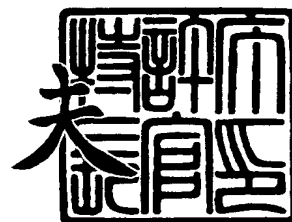
出      願      人            日 本 電 気 株 式 会 社  
Applicant(s):



2 0 0 3 年 1 0 月 1 7 日

特許庁長官  
Commissioner,  
Japan Patent Office

今 井 康 夫



【書類名】 特許願

【整理番号】 34002301

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/46  
G06F 15/16

【発明者】

【住所又は居所】 東京都港区芝五丁目 7 番 1 号 日本電気株式会社内

【氏名】 柴山 充文

【発明者】

【住所又は居所】 東京都港区芝五丁目 7 番 1 号 日本電気株式会社内

【氏名】 大澤 拓

【発明者】

【住所又は居所】 東京都港区芝五丁目 7 番 1 号 日本電気株式会社内

【氏名】 松下 智

【特許出願人】

【識別番号】 000004237

【氏名又は名称】 日本電気株式会社

【代理人】

【識別番号】 100088890

【弁理士】

【氏名又は名称】 河原 純一

【手数料の表示】

【予納台帳番号】 009690

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9001717

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プログラム並列化装置、プログラム並列化方法およびプログラム並列化プログラム

【特許請求の範囲】

【請求項 1】 逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、

前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、

与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、

前記フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、

前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部と

を有することを特徴とするプログラム並列化装置。

【請求項 2】 逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、

前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、

与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、

前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク 1

回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、

前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部と

を有することを特徴とするプログラム並列化装置。

【請求項 3】 前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とする請求項 1 または請求項 2 記載のプログラム並列化装置。

【請求項 4】 前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とする請求項 1、請求項 2 または請求項 3 記載のプログラム並列化装置。

【請求項 5】 前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とする請求項 4 記載のプログラム並列化装置。

【請求項 6】 前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実

行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする請求項1，請求項2または請求項3記載のプログラム並列化装置。

【請求項7】 前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに2つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とする請求項6記載のプログラム並列化装置。

【請求項8】 逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、  
逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、

制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定する工程と、  
決定されたフォーク箇所候補からフォーク箇所候補の試行組合せを生成する工程と、  
生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、  
評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、  
決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と  
を含むことを特徴とするプログラム並列化方法。

【請求項 9】 逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において

、  
逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、  
制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定する工程と、  
決定されたフォーク箇所候補からフォーク 1 回モデル上で互いに同時に実行可能なフォーク箇所候補のみからなる試行組合せを生成する工程と、  
生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、  
評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、  
決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と  
を含むことを特徴とするプログラム並列化方法。

【請求項 10】 制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、  
フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよび



データフローの解析結果を参照してフォーク箇所候補を生成する工程と、  
フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、  
フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、  
並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、  
並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、  
並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、  
フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を1つ選択する工程と、  
フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、  
フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、  
フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、  
フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、  
フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、  
フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、  
フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、  
フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工程と、  
フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として出力する工程と、  
並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と  
を含むことを特徴とするプログラム並列化方法。

【請求項 11】 制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、

フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照してフォーク箇所候補を生成する工程と、

フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、

フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、

並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、

並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、

並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、

フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を1つ選択する工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、

フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、選択されたフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、

フォーク箇所候補最良組合せ決定部が、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、

フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、

フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、

フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工程と、

フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部に出力する工程と、

並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程と

を含むことを特徴とするプログラム並列化方法。

【請求項 12】 前記並列実行性能を評価する工程で、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とする請求項 8，請求項 9，請求項 10 または請求項 11 記載のプログラム並列化方法。

【請求項 13】 前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とする請求項 8，請求項 9，請求項 10 または請求項 11 記載のプログラム並列化方法。

【請求項 14】 前記フォーク箇所候補の最良組合せを決定する工程で、全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク

箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能をある入力データに対して評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする請求項 8、請求項 9、請求項 10 または請求項 11 記載のプログラム並列化方法。

【請求項 15】 前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部の処理が終了すると、フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに 2 つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とする請求項 8、請求項 9、請求項 10 または請求項 11 記載のプログラム並列化方法。

【請求項 16】 コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部、前記制御・データフロー解析部

による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部、前記フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部、ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラム。

【請求項 17】 コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部、前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク 1 回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部、ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラム。

【請求項 18】 前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素

の単位でシミュレートして並列実行性能を算出することを特徴とする請求項 1 6 または請求項 1 7 記載のプログラム並列化プログラム。

【請求項 1 9】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とする請求項 1 6，請求項 1 7 または請求項 1 8 記載のプログラム並列化プログラム。

【請求項 2 0】前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とする請求項 1 9 記載のプログラム並列化プログラム。

【請求項 2 1】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする請求項 1 6，請求項 1 7 または請求項 1 8 記載のプログラム並列化プログラム。

【請求項 2 2】前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行

わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに2つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とする請求項21記載のプログラム並列化プログラム。

#### 【発明の詳細な説明】

#### 【0001】

#### 【発明の属する技術分野】

本発明は、複数のスレッドを複数のプロセッサにより並列に実行するマルチスレッド型プロセッサのためのプログラム変換技術に関し、特に実行効率の高い並列化プログラムを高速に生成するためのプログラム並列化装置、プログラム並列化方法およびプログラム並列化プログラム（コンパイラ）に関する。

#### 【0002】

#### 【従来の技術】

単一の逐次処理プログラムを並列プロセッサシステムで並列に処理する手法として、プログラムをスレッドと呼ぶ命令流に分割して複数のプロセッサで並列に実行するマルチスレッド実行方法がある（例えば、特許文献1～3，非特許文献1参照）。以下、これらの従来のマルチスレッド実行方法について説明する。

#### 【0003】

一般に、マルチスレッド実行方法において、他のプロセッサ上に新たなスレッド



を生成することを、スレッドをフォーク（f o r k）するといい、フォーク動作を行った側のスレッドを親スレッド、生成された新しいスレッドを子スレッド、スレッドをフォークするプログラム位置をフォーク点、子スレッドの先頭のプログラム位置をフォーク先アドレスまたは子スレッドの開始点と呼ぶ。特許文献1～3および非特許文献1では、スレッドのフォークを指示するためにフォーク点にフォーク命令が挿入される。フォーク命令にはフォーク先アドレスが指定され、フォーク命令の実行によりそのフォーク先アドレスから始まる子スレッドが他プロセッサ上に生成され、子スレッドの実行が開始される。また、スレッドの処理を終了させるプログラム位置をターム点と呼び、各プロセッサはスレッドの処理を終了する。

#### 【0004】

図15は、マルチスレッド実行方法の処理の概要を示す図である。同図（a）は、3つのスレッドA、B、Cに分割された逐次処理プログラムを示す。このプログラムを単一のプロセッサで処理する場合、同図（b）に示すように1つのプロセッサPEがスレッドA、B、Cを順番に処理していく。これに対して、特許文献1～3および非特許文献1のマルチスレッド実行方法では、同図（c）に示すように、1つのプロセッサPE1にスレッドAを実行させ、プロセッサPE1でスレッドAを実行している最中に、スレッドAに埋め込まれたフォーク命令によってスレッドBを他のプロセッサPE2に生成し、プロセッサPE2においてスレッドBを実行させる。また、プロセッサPE2は、スレッドBに埋め込まれたフォーク命令によってスレッドCをプロセッサPE3に生成する。プロセッサPE1、PE2は、それぞれスレッドB、Cの開始点の直前のターム点においてスレッドの処理を終了し、プロセッサPE3は、スレッドCの最後の命令を実行すると、その次の命令（一般にはシステムコール命令）を実行する。このように、複数のプロセッサでスレッドを同時に並行して実行することにより、逐次処理に比べて性能の向上が図られる。

#### 【0005】

他のマルチスレッド実行方法として、図15（d）に示すように、スレッドAを実行しているプロセッサPE1からフォークを複数回行うことにより、プロセッ

サPE 2にスレッドBを、またプロセッサPE 3にスレッドCをそれぞれ生成するマルチスレッド実行方法も存在する。図15 (d) のモデルに対して、同図(c) に示したように、スレッドはその生存中に高々1回に限って有効な子スレッドを生成することができるという制約を課したマルチスレッド実行方法を、フォーク1回モデルと呼ぶ。フォーク1回モデルでは、スレッド管理の大幅な簡略化が可能となり、現実的なハードウェア規模でスレッド管理部のハードウェア化が実現できる。また、個々のプロセッサは子スレッドを生成する他プロセッサが1プロセッサに限定されるため、隣接するプロセッサを単方向にリング状に接続した並列プロセッサシステムでマルチスレッド実行が可能となる。

#### 【0006】

親スレッドが子スレッドを生成し、子スレッドに所定の処理を行わせるには、親スレッドのフォーク点におけるレジスタファイル中のレジスタのうち少なくとも子スレッドに必要なレジスタの値を親スレッドから子スレッドに引き渡す必要がある。このスレッド間のデータ引き渡しコストを削減するために、非特許文献1および特許文献2では、スレッド生成時のレジスタ値継承機構をハードウェア的に備えている。これは、スレッド生成時に親スレッドのレジスタファイルの内容を子スレッドに全てコピーするものである。子スレッドの生成後は、親スレッドと子スレッドとのレジスタ値の変更は独立となり、レジスタを用いたスレッド間のデータの引き渡しは行われない。スレッド間のデータ引き渡しに関する他の従来技術としては、レジスタの値を命令によりレジスタ単位で個別に転送する機構を備えた並列プロセッサシステムも提案されている。

#### 【0007】

マルチスレッド実行方法では、実行の確定した先行スレッドを並列に実行することを基本とするが、実際のプログラムでは実行の確定するスレッドが十分に得られない場合も多い。また、動的に決定される依存やコンパイラ解析能力の限界等により並列化率が低く抑えられ、所望の性能が得られない可能性が生じる。このため、特許文献1では、制御投機を導入し、ハードウェア的にスレッドの投機実行をサポートしている。制御投機では、実行する可能性の高いスレッドを実行確定前に投機的に実行する。投機状態のスレッドは、実行の取り消しがハードウェア

ア上可能である範囲内で仮実行を行う。子スレッドが仮実行を行っている状態を仮実行状態といい、子スレッドが仮実行状態にあるとき親スレッドはスレッド仮生成状態にあるという。仮実行状態の子スレッドでは共有メモリおよびキャッシュメモリへの書き込みは抑制され、別途設けた仮実行用バッファ（temporary buffer）に対して書き込みが行われる。投機が正しいことが確定すると、親スレッドから子スレッドに対して投機成功通知が出され、子スレッドは仮実行用バッファの内容を共有メモリおよびキャッシュメモリに反映し、仮実行用バッファを用いない通常の状態となる。また、親スレッドは、スレッド仮生成状態からスレッド生成状態となる。他方、投機が失敗したことが確定すると、親スレッドでスレッド破棄命令（abort）が実行され、子スレッド以下の実行がキャンセルされる。また、親スレッドは、スレッド仮生成状態からスレッド未生成状態となり、再び子スレッドの生成が可能になる。つまり、フォーク1回モデルでは、スレッド生成は高々1回に限定されるが、制御投機を行い、投機が失敗した場合には再びフォークが可能となる。この場合においても、有効な子スレッドは高々1つである。

#### 【0008】

スレッドはその生存中に高々1回に限って有効な子スレッドを生成するというフォーク1回モデルのマルチスレッド実行を実現するために、例えば非特許文献1等では、逐次処理プログラムから並列化プログラムを生成するコンパイルの段階で、全てのスレッドが有効なフォークを1回しか実行しない命令コードになるように制限している。すなわち、フォーク1回制限を並列化プログラム上において静的に保証している。

#### 【0009】

一方、特許文献3では、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する1つのフォーク命令を親スレッドの実行中に選択することにより、フォーク1回制限をプログラム実行時に保証している。

#### 【0010】

次に、上述したマルチスレッド実行方法で実行される従来の並列化プログラム生

成装置について説明する。

【0011】

図16を参照すると、従来のプログラム並列化装置10は、逐次処理プログラム13を入力し、制御・データフロー解析部11によって逐次処理プログラム13の制御フローおよびデータフローを解析し、次いでその結果に基づき、フォーク挿入部12において、基本ブロックあるいは複数の基本ブロックを並列化の単位、すなわちスレッドに分割し、次いで並列化のためのフォーク命令を挿入して、複数のスレッドに分割された並列化プログラム14を生成し出力する。

【0012】

フォーク命令の挿入箇所（フォーク箇所）は、制御フローおよびデータフローの解析結果を参照して、より並列実行性能が得られるように決定される。しかし、プログラム実行時にしか判明しないプログラム実行フローやメモリ依存関係などの影響により、静的な解析に基づいたフォーク命令の挿入方法では所望の並列実行性能を得られない場合がある。

【0013】

これに対し、逐次実行を行ったときの条件分岐確率やデータ依存発生頻度などのプロファイル情報を参照して、フォーク箇所を決定する方法が公知である（例えば、特許文献4参照）。この方法によれば、逐次実行時の動的な情報であるプロファイル情報を利用することで、より適切なフォーク命令の挿入が可能となり、並列実行性能の向上が期待できる。

【0014】

【特許文献1】

特開10-27108号公報（第5頁、図1）

【非特許文献1】

「On Chip Multiprocessor指向 制御並列アーキテクチャMUSCATの提案」（並列処理シンポジウムJSP97論文集、情報処理学会、pp. 229-236、May 1997）

【特許文献2】

特開10-78880号公報（第7頁、図2）

**【特許文献 3】**

特開 2 0 0 3 - 0 2 9 9 8 5 号公報（第 1 3 頁、図 1 0）

**【特許文献 4】**

特開 2 0 0 1 - 2 8 2 5 4 9 号公報（第 1 1 - 1 2 頁、図 2）

**【0 0 1 5】****【発明が解決しようとする課題】**

ところで、上記した従来の技術は、下記の問題点を有している。

**【0 0 1 6】**

第 1 の問題点は、フォーク箇所を決定する基準にまだ多分に改良の余地があるため、従来の技術をもってしても、所望の並列実行性能を得られない場合があることである。その理由は、フォーク箇所を決定する基準は、望ましくはそのフォーク命令の並列実行性能への寄与度であるが、たとえ逐次実行時のプロファイル情報を利用したとしても、並列実行時の性能を解析的な方法で精度良く予測することは困難であるため、適切にフォーク命令を挿入できない場合があるからである。

**【0 0 1 7】**

並列実行性能を解析的に高精度に予測することを困難にしている要因の一つに、並列実行性能に対するメモリ依存関係の影響がある。並列実行性能に対するメモリ依存関係の影響は、並列化の仕方に応じて複雑に変化するため、逐次実行時のプロファイル情報からメモリ依存関係の情報を得たとしても、その並列実行性能への影響を精度良く評価することは困難である。また、フォーク 1 回モデルなどの並列化方法上の制約や、プロセッサ間接続方式などのハードウェア構成等も並列実行性能に大きな影響を与える。しかし、同様に、これらの並列実行性能への影響は、並列化の仕方に応じて複雑に変化するため、逐次実行時のプロファイル情報から解析的な方法により精度良く評価することは困難である。

**【0 0 1 8】**

すなわち、制御フローおよびデータフローの解析結果や逐次実行時に得られるプロファイル情報からでは、並列実行性能を精度良く予測することは困難であるため、適切にフォーク命令を挿入できず、所望の並列実行性能を得られない場合が

あるという問題がある。

【0019】

第2の問題点は、より良い並列実行性能のフォーク箇所を得ようとするにつれて、そのフォーク箇所の決定処理にますます時間がかかるということである。その理由は、フォーク箇所を決定する評価基準を高精度化するにつれて、その評価処理にますます時間がかかるのに加えて、フォーク箇所の数は、通常、例えばプログラムに含まれるループ数などと比較しても多く、従ってその組合せの数は膨大になるからである。

【0020】

本発明は、このような事情に鑑みて提案されたものであり、その目的は、フォーク箇所候補最良組合せ決定処理を高精度かつ高速に行い、並列実行性能のより高い並列化プログラムを高速に生成するプログラム並列化装置、プログラム並列化方法およびプログラム並列化プログラムを提供することにある。

【0021】

【課題を解決するための手段】

前記目的を達成するため、本発明のプログラム並列化装置、プログラム並列化方法およびプログラム並列化プログラムは、入力される逐次処理プログラムの制御フローおよびデータフローの解析結果や逐次実行時に得られるプロファイル情報に加えて、並列実行性能を直接評価した結果を基準に、フォーク箇所候補の最良組合せを決定することを特徴とする。

【0022】

具体的には、第1の発明は、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、前記フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列

実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部とを有することを特徴とするプログラム並列化装置である。

#### 【0023】

また、第2の発明は、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部と、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部と、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部と、前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク1回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部と、前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部とを有することを特徴とするプログラム並列化装置である。

#### 【0024】

さらに、第3の発明は、前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とするプログラム並列化装置である。

#### 【0025】

さらにまた、第4の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フ

オーク箇所候補決定部により決定されたオーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のオーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とするプログラム並列化装置である。

**【0026】**

また、第5の発明は、前記オーク箇所候補最良組合せ決定部が、求められたオーク箇所候補の順序で上位から一定数を含むオーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのオーク箇所候補の最良組合せとすることを特徴とするプログラム並列化装置である。

**【0027】**

さらに、第6の発明は、前記オーク箇所候補最良組合せ決定部が、前記オーク箇所候補決定部により決定された全オーク箇所候補の集合を互いになるべく影響を及ぼし合わないようオーク箇所候補のグループに分割し、分割されたオーク箇所候補のグループの中でオーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してオーク箇所候補の試行組合せを生成し、それによる並列実行性能を前記並列実行性能評価部によって評価した結果を参照してオーク箇所候補の最良組合せを決定するオーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるオーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とするプログラム並列化装置である。

**【0028】**

さらにまた、第7の発明は、前記オーク箇所候補最良組合せ決定部が、前記オーク箇所候補決定部の処理が終了すると、前記オーク箇所候補決定部により決定された全オーク箇所候補を集合としてオーク箇所候補のグループ分割処理を呼び出し、オーク箇所候補のグループ分割処理が呼び出されると、与えられたオーク箇所候補の集合に属するオーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにオーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするオーク箇所候補の数が指定数より大きいフォー



ク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに2つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とするプログラム並列化装置である。

#### 【0029】

また、第8の発明は、逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定する工程と、決定されたフォーク箇所候補からフォーク箇所候補の試行組合せを生成する工程と、生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

#### 【0030】

さらに、第9の発明は、逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法のためのプログラム並列化方法において、逐次処理プログラムの制御フローおよびデータフローの解析を行う工程と、制御フローおよびデータフローの解析結果を参照して逐次処理プログラ

ムのフォーク箇所候補を決定する工程と、決定されたフォーク箇所候補からフォーク 1 回モデル上で互いに同時に実行可能なフォーク箇所候補のみからなる試行組合せを生成する工程と、生成されたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する工程と、評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定する工程と、決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

### 【0031】

さらにまた、第 10 の発明は、制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照してフォーク箇所候補を生成する工程と、フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を 1 つ選択する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最

良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として出力する工程と、並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

#### 【0032】

また、第11の発明は、制御・データフロー解析部が、逐次処理プログラムの制御フローおよびデータフローを解析する工程と、フォーク箇所候補決定部が、制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照してフォーク箇所候補を生成する工程と、フォーク箇所候補最良組合せ決定部が、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける工程と、フォーク箇所候補最良組合せ決定部が、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする工程と、並列実行性能評価

部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成する工程と、並列実行性能評価部が、逐次実行トレースをすべてのターム点候補を分割点として分割する工程と、並列実行性能評価部が、各スレッド要素毎のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する工程と、フォーク箇所候補最良組合せ決定部が、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を1つ選択する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、フォーク箇所候補最良組合せ決定部が、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補の最良組合せから選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする工程と、フォーク箇所候補最良組合せ決定部が、試行組合せによる並列化の並列実行性能を並列実行性能評価部によって評価する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する工程と、フォーク箇所候補最良組合せ決定部が、試行組合せの並列実行性能の方が良ければ、選択されたフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから取り除く工程と、フォーク箇所候補最良組合せ決定部が、試行組合せを現時点でのフォーク箇所候補の最良組合せとする工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補があるかどうかを判定し、未選択のフォーク箇所候補があれば繰り返し実行する工程と、フォーク箇所候補最良組合せ決定部が、未選択のフォーク箇所候補がなければ、直前の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する工程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つかった場合には、すべてのフォーク箇所候補を未選択の状態にして繰り返し実行する工

程と、フォーク箇所候補最良組合せ決定部が、新たにフォーク箇所候補の最良組合せが見つからなかった場合には、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部に出力する工程と、並列化プログラム出力部が、フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する工程とを含むことを特徴とするプログラム並列化方法である。

#### 【0033】

さらに、第12の発明は、前記並列実行性能を評価する工程で、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とするプログラム並列化方法である。

#### 【0034】

さらにまた、第13の発明は、前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とするプログラム並列化方法である。

#### 【0035】

また、第14の発明は、前記フォーク箇所候補の最良組合せを決定する工程で、全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能をある入力データに対して評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とする

プログラム並列化方法である。

#### 【0036】

さらに、第15の発明は、前記フォーク箇所候補の最良組合せを決定する工程で、前記フォーク箇所候補決定部の処理が終了すると、フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元に戻り、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに2つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元に戻り、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とするプログラム並列化方法である。

#### 【0037】

さらにまた、第16の発明は、コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部、前記

フォーク箇所候補決定部により決定されたフォーク箇所候補の試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部、ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラムである。

#### 【0038】

また、第17の発明は、コンピュータを、逐次処理プログラムの制御フローおよびデータフローの解析を行う制御・データフロー解析部、前記制御・データフロー解析部による制御フローおよびデータフローの解析結果を参照して逐次処理プログラムのフォーク箇所候補を決定するフォーク箇所候補決定部、与えられたフォーク箇所候補の試行組合せにより逐次処理プログラムを並列化したときの並列実行性能を、ある入力データに対して評価する並列実行性能評価部、前記フォーク箇所候補決定部により決定されたフォーク箇所候補からフォーク1回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを前記並列実行性能評価部に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部、ならびに前記フォーク箇所候補最良組合せ決定部により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラムを生成し出力する並列化プログラム出力部として動作させることを特徴とするプログラム並列化プログラムである。

#### 【0039】

さらに、第18の発明は、前記並列実行性能評価部が、逐次処理プログラムを入力データで逐次実行したときの逐次実行トレースを生成し、逐次実行トレースをすべてのターム点候補を分割点として分割し、各スレッド要素毎のスレッド要素情報を解析し、与えられたフォーク箇所候補の試行組合せに対して並列実行をスレッド要素の単位でシミュレートして並列実行性能を算出することを特徴とするプログラム並列化プログラムである。

**【0040】**

さらにまた、第19の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定されたフォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくことを特徴とするプログラム並列化プログラムである。

**【0041】**

また、第20の発明は、前記フォーク箇所候補最良組合せ決定部が、求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を前記並列実行性能評価部によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとすることを特徴とするプログラム並列化プログラムである。

**【0042】**

さらに、第21の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を前記並列実行性能評価部によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定することを特徴とするプログラム並列化プログラムである。

**【0043】**

さらにまた、第22の発明は、前記フォーク箇所候補最良組合せ決定部が、前記フォーク箇所候補決定部の処理が終了すると、前記フォーク箇所候補決定部により決定された全フォーク箇所候補を集合としてフォーク箇所候補のグループ分割処理を呼び出し、フォーク箇所候補のグループ分割処理が呼び出されると、与えられたフォーク箇所候補の集合に属するフォーク箇所候補の数が指定された分割



下限数より大きければ、集合のグループ分割処理を開始し、小さければ、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元にリターンし、自身をキャンセルするフォーク箇所候補の数が指定数より大きいフォーク箇所候補の集合を集合から分割して新たにグループを生成し、集合をさらに2つのグループにグループ分割し、一方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して一方のグループのグループ分割を行い、他方のグループを集合としてフォーク箇所候補のグループ分割処理を再帰的に呼び出して他方のグループのグループ分割を行い、フォーク箇所候補のグループ分割処理を呼び出した元にリターンし、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行い、全グループの処理が完了したかどうかを判定し、未処理のグループがあるならば、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返し、全グループの処理が完了した場合は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力することを特徴とするプログラム並列化プログラムである。

#### 【0044】

本発明によれば、フォーク箇所候補の試行組合せに対して並列実行性能を直接評価し、それを基準にフォーク箇所候補の最良組合せを決定するので、フォーク命令を逐次処理プログラムにより適切に挿入することが可能であり、並列実行性能のより高い並列化プログラムを得ることができる。

#### 【0045】

また、本発明によれば、フォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくため、フォーク箇所候補のすべての組合せを評価する場合に比べて大幅にフォーク箇所候補最良組合せ決定処理の処理時間を短縮することができるという効果がある。

#### 【0046】

さらに、本発明によれば、マルチスレッド実行方法としてフォーク1回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを

対象に並列実行性能を評価するので、フォーク箇所候補最良組合せ決定処理の処理時間を短縮することができるという効果がある。

**【0047】**

さらにまた、本発明によれば、フォーク箇所候補をグループ間では互いになるべく影響を及ぼし合わないよう適当数のグループに分割して、各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うので、フォーク箇所候補の組合せの総数を削減することができ、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができるという効果がある。

**【0048】**

また、本発明によれば、ターム点候補で決定されるスレッド要素の単位で並列実行のシミュレーションを行うので、並列実行性能を高速に評価することが可能であり、従ってフォーク箇所候補最良組合せ決定処理を高速に行うことができるという効果がある。

**【0049】**

図3を参照して具体例を説明する。図3は、隣接するプロセッサを単方向にリング状に接続した並列プロセッサシステム上で、フォーク1回モデルによるマルチスレッド実行の概略を示す図である。同図(a)は、3つのスレッド $A_n$ 、 $B_n$ 、 $C_n$  ( $n = \dots, 1, 2, 3, \dots$ ) が繰り返し実行される逐次処理プログラムの逐次実行トレースを示す。ここで、フォーク箇所候補として、スレッド $A_n$ の先頭をフォーク点、スレッド $A_n$ の終端をフォーク先とするフォーク箇所候補 $f_0$ と、スレッド $B_n$ の先頭をフォーク点、スレッド $B_n$ の終端をフォーク先とするフォーク箇所候補 $f_1$ と、スレッド $B_n$ の先頭をフォーク点、スレッド $C_n$ の終端をフォーク先とするフォーク箇所候補 $f_2$ とがあるとする。

**【0050】**

3つのフォーク箇所候補 $f_0$ 、 $f_1$ 、 $f_2$ の組合せは、 $\phi$  (フォークしない)、 $\{f_0\}$ 、 $\{f_1\}$ 、 $\{f_2\}$ 、 $\{f_0, f_1\}$ 、 $\{f_0, f_2\}$ 、 $\{f_1, f_2\}$ 、 $\{f_0, f_1, f_2\}$  の8通り存在する。フォーク箇所候補最良組合せ決定処理は、フォーク箇所候補の組合せの中からより良い並列実行性能を示す組合せを選択する処理に他ならない。仮にいずれのフォーク箇所候補の組合せを選択

してもデータ依存関係を壊さないとする、図3 (b) は {f 0} により並列実行した（フォーク箇所候補 f 0 のみにフォーク命令を挿入した）ときの様子を示す図である。同様に、同図 (c) は {f 1} により、同図 (d) は {f 2} により、同図 (e) は {f 0, f 1} により、同図 (f) は {f 0, f 2} により、並列実行したときの様子を示す図である。

#### 【0051】

いま、フォーク1回モデルを仮定しているので、フォーク箇所候補 f 1 とフォーク箇所候補 f 2 とは同時に実行することができず、{f 1, f 2} および {f 0, f 1, f 2} はそのまま並列実行することができない。従って、非特許文献1等で示されているように並列化コードの生成時に高々どちらか一方のみが挿入されるように保証するか、または特許文献3で示されているように実行時にどちらか一方のみが選択されて実行されることになる。

#### 【0052】

図3 (a) に示した逐次実行した場合を基準にすると、同図 (b) の {f 0} による並列化は1.33倍の処理量を提供しているため、逐次実行性能の1.33倍の並列実行性能を達成している。すなわち、逐次実行に対して1.33倍プログラム実行が高速である。同様に、同図 (c) の {f 1} による並列化では2.67倍、同図 (d) の {f 2} による並列化では4倍、同図 (e) の {f 0, f 1} による並列化では3.2倍、同図 (f) の {f 0, f 2} による並列化では2.67倍の並列実行性能を達成している。すなわち、本具体例では、例えばフォーク箇所候補 f 0 とフォーク箇所候補 f 1 または f 2 とを組み合わせるよりも、フォーク箇所候補 f 2 単独とした方がより良い並列実行性能を示す。これは、フォーク1回制限やプロセッサが単方向にリング状に接続されているというハードウェア構成も大きく影響しており、さらにデータ依存関係に違反するような状況ではより複雑な様相を呈する。このため、逐次処理プログラムの制御フローおよびデータフローの解析結果や逐次実行時に得られるプロファイル情報のみからでは、このような並列実行性能を予測することは困難である。従って、従来の並列化方法では所望の並列実行性能を得られない場合があったのに対して、本発明によれば、並列実行性能を直接評価し、それを基準にしてフォーク箇所候補の最

良組合せを決定するので、フォーク命令を逐次処理プログラムにより適切に挿入することが可能であり、並列実行性能のより高い並列化プログラムを得ることができる。

#### 【0053】

##### 【発明の実施の形態】

以下、本発明の実施の形態について図面を参照して詳細に説明する。

#### 【0054】

##### 〔第1の実施の形態〕

図1を参照すると、本発明の第1の実施の形態に係るプログラム並列化装置20は、入力される逐次処理プログラム13の制御フローおよびデータフローの解析を行う制御・データフロー解析部21と、制御・データフロー解析部21による制御フローおよびデータフローの解析結果を参照して逐次処理プログラム13のフォーク箇所候補を決定するフォーク箇所候補決定部22と、フォーク箇所候補決定部22により決定されたフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を並列実行性能評価部24によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部23と、フォーク箇所候補最良組合せ決定部23より与えられるフォーク箇所候補の試行組合せにより逐次処理プログラム13を並列化したときの並列実行性能を、ある入力データ15に対して評価する並列実行性能評価部24と、フォーク箇所候補最良組合せ決定部23により決定されたフォーク箇所候補の最良組合せに基づいて並列化プログラム14を生成し出力する並列化プログラム出力部25とを備える。

#### 【0055】

図2を参照すると、フォーク箇所候補決定部22，フォーク箇所候補最良組合せ決定部23，および並列実行性能評価部24によるフォーク箇所候補最良組合せ決定処理は、フォーク箇所候補生成ステップ100と、フォーク箇所候補順序付けステップ101と、フォーク箇所候補初期組合せ生成ステップ102と、逐次実行トレース生成ステップ103と、スレッド要素分割ステップ104と、スレッド要素情報解析・記憶ステップ105と、フォーク箇所候補選択ステップ10

6 と、フォーク箇所候補最良組合せ判定ステップ 107 と、フォーク箇所候補加入・試行組合せ設定ステップ 108 と、フォーク箇所候補削除・試行組合せ設定ステップ 109 と、並列実行性能算出ステップ 110 と、並列実行性能比較ステップ 111 と、最良組合せ設定ステップ 112 と、全フォーク箇所候補選択判定ステップ 113 と、新最良組合せ発見判定ステップ 114 と、全フォーク箇所候補未選択設定ステップ 115 と、最良組合せ出力ステップ 116 とからなる。

#### 【0056】

次に、このように構成された第 1 の実施の形態に係るプログラム並列化装置 20 の動作について説明する。

#### 【0057】

特に、図 2 を参照して、フォーク箇所候補決定部 22，フォーク箇所候補最良組合せ決定部 23，および並列実行性能評価部 24 で処理されるフォーク箇所候補最良組合せ決定処理を詳細に説明する。

#### 【0058】

まず、制御・データフロー解析部 21 は、逐次処理プログラム 13 の制御フローおよびデータフローを解析する。

#### 【0059】

次に、フォーク箇所候補決定部 22 は、制御・データフロー解析部 21 による制御フローおよびデータフローの解析結果を参照して、フォーク箇所候補を生成する（ステップ 100）。具体的には、制御フローを解析して逐次処理プログラム 13 の基本ブロックを抽出し、その基本ブロックあるいは複数の基本ブロックを単位として並列化を行うようにフォーク箇所候補を生成する。このとき、例えば、非特許文献 1 や特許文献 2 に開示されているスレッド生成時にレジスタ値を子スレッドに継承し、子スレッド生成後は親スレッドと子スレッド間でレジスタを用いたデータの引き渡しを行わないマルチスレッド実行方法では、データフロー上でレジスタの内容が同じである任意の箇所でフォークを行うことが可能であり、フォーク箇所候補となり得る。また、非特許文献 1（4. 1 節）に開示されている方針などにより、制御フローおよびデータフローを参照して並列化性能に対して有効だと予測されるフォーク箇所候補をある程度あらかじめ絞り込んでもよ

い。

#### 【0 0 6 0】

次に、フォーク箇所候補最良組合せ決定部 2 3 は、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける（ステップ 1 0 1）。効果の予測は、例えば、逐次実行したときの逐次実行トレースの結果を参照して、予測対象のフォーク箇所候補のフォーク点からフォーク先までの実行サイクル数や該フォーク箇所候補の出現回数を参照して効果を予測してもよいし、フォーク箇所候補をそれぞれ単独で並列実行したときの並列実行性能を並列実行性能評価部 2 4 によって評価して効果を求めてもよい。

#### 【0 0 6 1】

続いて、フォーク箇所候補最良組合せ決定部 2 3 は、最初に並列実行性能を評価するフォーク箇所候補の組合せ（初期組合せ）を生成し、それをフォーク箇所候補の最良組合せとする（ステップ 1 0 2）。これは、例えば、ステップ 1 0 1 で求めたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を並列実行性能評価部 2 4 によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとする。

#### 【0 0 6 2】

次に、並列実行性能評価部 2 4 は、まず、逐次処理プログラム 1 3 を入力データ 1 5 で逐次実行したときの逐次実行トレースを生成する（ステップ 1 0 3）。

#### 【0 0 6 3】

続いて、並列実行性能評価部 2 4 は、逐次実行トレースをすべてのターム点候補を分割点として分割する（ステップ 1 0 4）。分割された各々の部分実行トレースを、スレッド要素と呼ぶ。ターム点候補は、フォーク箇所候補が与えられれば決定することができる。例えば、特許文献 3 に開示されている、有効な子スレッドを生成したプロセッサが有効な子スレッドの開始アドレスの直前のアドレスの命令まで実行を完了するマルチスレッド実行方法（特許文献 3 の請求項 1，段落 0 0 1 3 等参照）では、ターム点はフォーク命令のフォーク先に等しく、フォー

ク箇所候補を与えればターム点候補は一意に決定される。

#### 【0064】

次に、並列実行性能評価部24は、各スレッド要素毎の実行サイクル数、スレッド要素内に出現するフォーク箇所候補とその出現位置、フォーク先のスレッド要素、スレッド要素間のメモリ依存関係等のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する（ステップ105）。

#### 【0065】

例えば、図3（a）を参照すると、逐次実行トレースは、フォーク箇所候補  $f_0$ 、 $f_1$ 、 $f_2$  によって、スレッド要素  $A_n$ 、 $B_n$ 、 $C_n$  に分割することができる。ここで、スレッド要素  $A_n$  およびスレッド要素  $B_n$  は、フォーク箇所候補  $f_0$  のターム点により、スレッド要素  $B_n$  およびスレッド要素  $C_n$  はフォーク箇所候補  $f_1$  のターム点により、スレッド要素  $C_n$  およびスレッド要素  $A_n$  はフォーク箇所候補  $f_2$  のターム点により決定される。

#### 【0066】

フォーク箇所候補  $f_0$ 、 $f_1$ 、 $f_2$  で生成可能なスレッドは、すべてスレッド要素  $A_n$ 、 $B_n$ 、 $C_n$  の組合せで構成される。例えば、図3（b）に示すフォーク箇所候補  $f_0$  で生成されるスレッドは、スレッド要素  $B_n$ 、 $C_n$ 、 $A_n$  の順序で構成されている。同様に、図3（c）～（f）に示すフォーク箇所候補の他の組合せによる並列化におけるスレッドも、すべてスレッド要素  $A_n$ 、 $B_n$ 、 $C_n$  の組合せで構成される。これは、フォーク箇所候補が与えられたとき、そのフォーク箇所候補のターム点で決定されるスレッド要素の単位で、与えられたフォーク箇所候補による並列実行をシミュレートすることが可能であることを意味する。しかしながら、各スレッドの開始位置（スレッド内のフォークを実行する位置）の決定やデータ依存関係の検出のためには、逐次実行トレースを命令単位で解析する必要がある。本実施の形態では、与えられた逐次処理プログラム13および入力データ15に対して、各スレッド要素毎の実行サイクル数、スレッド要素内に出現するフォーク箇所候補とその出現位置、フォーク先のスレッド要素、スレッド要素間のメモリ依存度等のスレッド要素情報を一度だけ解析し、スレッド情報を各スレッド要素毎に記憶する。実際に並列実行をシミュレートする処理では

、命令単位での解析を必要としないので、並列実行性能を繰り返し評価する場合に評価速度が高速であるという特徴がある。

#### 【0 0 6 7】

続いて、フォーク箇所候補最良組合せ決定部 2 3 は、ステップ 1 0 6 からステップ 1 1 3 までで構成されるループ A を繰り返し実行中に、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を 1 つ選択する（ステップ 1 0 6）。

#### 【0 0 6 8】

次に、フォーク箇所候補最良組合せ決定部 2 3 は、ステップ 1 0 6 で選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する（ステップ 1 0 7）。

#### 【0 0 6 9】

選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、フォーク箇所候補最良組合せ決定部 2 3 は、ステップ 1 0 6 で選択されたフォーク箇所候補をフォーク箇所候補の最良組合せに加え、そのフォーク箇所候補の組合せを試行組合せとする（ステップ 1 0 8）。

#### 【0 0 7 0】

一方、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補最良組合せ決定部 2 3 は、フォーク箇所候補の最良組合せからステップ 1 0 6 で選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする（ステップ 1 0 9）。

#### 【0 0 7 1】

次に、フォーク箇所候補最良組合せ決定部 2 3 は、試行組合せによる並列化の並列実行性能を、並列実行性能評価部 2 4 によって評価する（ステップ 1 1 0）。

#### 【0 0 7 2】

並列実行性能評価部 2 4 による並列実行性能の評価方法として、マルチプロセッサシステム上で実際に実行して評価する方法や、マルチプロセッサシステムのシミュレータ（図示せず）上で並列実行のシミュレーションを行って評価する方法、詳しくは、並列実行性能を評価するフォーク箇所候補の試行組合せで逐次処理



プログラム 13 を並列化したとして、そのときの動作をシミュレータ上で仮想的に実行して性能を評価する方法がある。また、このとき、並列実行性能の指針としては、並列実行したときの実行サイクル数や実行時間が適当であるが、消費電力を考慮するためにプロセッサ稼働率などを評価結果算出に加味してもよい。第 1 の実施の形態では、並列実行性能評価部 24 は、与えられたフォーク箇所候補の試行組合せに対して、ステップ 105 で記憶されたスレッド要素情報を参照して、ターム点候補で決定されるスレッド要素の単位で並列実行のシミュレーションを行って並列実行性能を評価する。

#### 【0073】

具体的には、並列実行性能評価部 24 は、まず、並列実行性能を評価したいフォーク箇所候補の試行組合せおよび各スレッド要素毎に記憶されているフォーク箇所候補のスレッド要素情報（スレッド要素内に出現するフォーク箇所候補、フォーク先のスレッド要素）を参照して自スレッドのターム点を決定し、スレッド要素を組み合わせで自スレッドを生成する。次に、並列実行性能評価部 24 は、フォーク箇所候補のスレッド要素情報（スレッド要素内に出現するフォーク箇所候補の出現位置）およびプロセッサ使用状況を参照して、そのフォーク箇所候補が生成する子スレッドの実行開始点を決定する。続いて、並列実行性能評価部 24 は、各スレッド要素毎に記憶されているフォーク箇所候補のスレッド要素情報（スレッド要素間のメモリ依存関係の情報）を参照してスレッド間のメモリ依存に対する投機実行の正否を判定し、実行の取消しがあればそれをスレッドの実行開始点に反映する。

#### 【0074】

並列実行性能評価部 24 によれば、ステップ 104 およびステップ 105 は、ある逐次実行トレースおよびターム点候補に対してあらかじめ 1 回実行しておけばよく、フォーク箇所候補最良組合せ決定処理のように並列実行性能を評価したいフォーク箇所候補の試行組合せを繰り返し与えられる場合には、ステップ 106 以降を実行すれば十分である。従って、例えば命令単位で並列化やメモリ依存のシミュレーションを評価毎に行う従来の並列実行シミュレーションに比べて、評価速度が高速であるという特徴がある。また、実際にマルチプロセッサ上で並列

実行を行って評価する方法にも、実働するマルチプロセッサがないと評価できない、評価したい試行組合せをマルチプロセッサで実行できる並列化プログラム 1 4 に変換して評価結果を取り出すのにオーバヘッドが存在する等の欠点があるため、本実施形態による並列実行性能の評価方法が評価処理の高速化に有効な場合が多い。

**【 0 0 7 5 】**

続いて、フォーク箇所候補最良組合せ決定部 2 3 は、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する（ステップ 1 1 1）。

**【 0 0 7 6 】**

試行組合せの並列実行性能の方が良ければ、フォーク箇所候補最良組合せ決定部 2 3 は、試行組合せを現時点でのフォーク箇所候補の最良組合せとする（ステップ 1 1 2）。試行組合せの並列実行性能の方が悪ければ、ステップ 1 1 2 をスキップする。

**【 0 0 7 7 】**

次に、フォーク箇所候補最良組合せ決定部 2 3 は、ループ A の繰り返し実行において、未選択のフォーク箇所候補があるかどうかを判定する（ステップ 1 1 3）。未選択のフォーク箇所候補があれば、ステップ 1 0 6 に制御を戻し、ループ A として繰り返し実行する。

**【 0 0 7 8 】**

未選択のフォーク箇所候補がなければ、すなわちすべてのフォーク箇所候補を選択したならば、フォーク箇所候補最良組合せ決定部 2 3 は、直前のループ A の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する（ステップ 1 1 4）。

**【 0 0 7 9 】**

新たにフォーク箇所候補の最良組合せが見つかった場合には、フォーク箇所候補最良組合せ決定部 2 3 は、ループ A の繰り返し実行に関して、すべてのフォーク箇所候補を未選択の状態にして（ステップ 1 1 5）、ステップ 1 0 6 に制御を戻し、ループ B として繰り返し実行する。

**【 0 0 8 0 】**

新たにフォーク箇所候補の最良組合せが見つからなかった場合には、フォーク箇所候補最良組合せ決定部 23 は、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部 25 に出力する（ステップ 116）。

#### 【0081】

最後に、並列化プログラム出力部 25 は、フォーク箇所候補最良組合せ決定部 23 により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラム 14 を生成し出力する。

#### 【0082】

以上説明したように、第 1 の実施の形態によれば、フォーク箇所候補の試行組合せに対して並列実行性能を直接評価し、それを基準にフォーク箇所候補の最良組合せを決定するので、フォーク命令を逐次処理プログラム 13 により適切に挿入することが可能であり、並列実行性能のより高い並列化プログラム 14 を得ることができる。

#### 【0083】

また、フォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序によって、その時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくため、フォーク箇所候補のすべての組合せを評価する場合に比べて、フォーク箇所候補最良組合せ決定処理の処理時間を大幅に短縮できるという効果がある。

#### 【0084】

[第 1 の実施の形態の具体例]

図 4 および図 5 を参照して、第 1 の実施の形態に係るプログラム並列化装置 20 におけるフォーク箇所候補最良組合せ決定処理の具体例を説明する。

#### 【0085】

ここで、フォーク箇所候補として  $f_0$ 、 $f_1$ 、 $f_2$ 、 $f_3$  があるとする。このときのフォーク箇所候補の組合せの総数は、まったくフォークをせずに逐次実行した場合 ( $\phi$ ) を含めて 16 通り存在する。

#### 【0086】

並列実行によって、逐次実行したとき（ $\phi$ の場合）の性能の何倍の性能が達成されるかを並列実行性能の指標として採用する。ここで、16通りのフォーク箇所候補の組合せのそれぞれの並列実行性能を、図4に示すように仮定する。この場合、フォーク箇所候補としてf1およびf2を選択したときの並列実行性能が最も高く、逐次実行の3.8倍の性能である。

#### 【0087】

フォーク箇所候補の組合せは16通りあるので、すべての組合せの並列実行性能を評価してフォーク箇所候補の最良組合せを決定する方法では、並列実行性能の評価は16回行う必要がある。

#### 【0088】

次に、図4に示す具体例に対する第1の実施の形態によるフォーク箇所候補最良組合せ決定処理の動作の詳細を、図2のフローチャートも参照しつつ説明する。

#### 【0089】

まず、ステップ100において、フォーク箇所候補としてf0, f1, f2, f3が得られているものとする。

#### 【0090】

次に、ステップ101において、フォーク箇所候補に順序を付ける。ここで、フォーク箇所候補それぞれを単独で並列実行したときの並列実行性能を指標として、それぞれの並列実行性能が大きい順序に順序付けを行うとする。図4に示すそれぞれの並列実行性能を参照すると、f0, f2, f1, f3の順序になる。

#### 【0091】

続いて、ステップ102において、初期組合せを生成する。ステップ101で求めたフォーク箇所候補の順序の上位から2つを初期組合せに採用すると、{f0, f2}が初期組合せになる。{f0, f2}の並列実行性能が評価され、その結果である3.2が得られる。また、{f0, f2}は、この時点でのフォーク箇所候補の最良組合せになる。

#### 【0092】

次に、ステップ106からループAおよびループBが繰り返し実行される。各繰り返しにおける動作の様子を、図5に示す。

**【0093】**

・ ループA実行1回目、ループB実行1回目（繰り返し1-1）

まず、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ106）。この時点では、すべてのフォーク箇所候補が未選択なので、フォーク箇所候補 f 0 が選択される。フォーク箇所候補 f 0 はフォーク箇所候補の最良組合せ {f 0, f 2} に含まれているので（ステップ107）、フォーク箇所候補の最良組合せ {f 0, f 2} からフォーク箇所候補 f 0 を取り除き、試行組合せは {f 2} となる（ステップ109）。次に、試行組合せの並列実行性能が評価されて、その結果である 2. 4 が得られる（ステップ110）。試行組合せの並列実行性能（2. 4）はフォーク箇所候補の最良組合せの並列実行性能（3. 2）よりも悪いので（ステップ111）、フォーク箇所候補の最良組合せは {f 0, f 2} のままである。ループAにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ113）、次にステップ106からループAの2回目の実行を開始する。

**【0094】**

・ ループA実行2回目、ループB実行1回目（繰り返し1-2）

同様にして、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ106）。この時点で、フォーク箇所候補 f 0 が選択済みなので、フォーク箇所候補 f 2 が選択される。フォーク箇所候補 f 2 はフォーク箇所候補の最良組合せ {f 0, f 2} に含まれているので（ステップ107）、フォーク箇所候補の最良組合せ {f 0, f 2} からフォーク箇所候補 f 2 を取り除き、試行組合せは {f 0} となる（ステップ109）。次に、試行組合せの並列実行性能が評価されて、その結果である 2. 5 が得られる（ステップ110）。試行組合せの並列実行性能（2. 5）はフォーク箇所候補の最良組合せの並列実行性能（3. 2）よりも悪いので（ステップ111）、フォーク箇所候補の最良組合せは {f 0, f 2} のままである。ループAにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ113）、次にステップ106からループAの3回目の実行を開始する。

**【0095】**

・ループA実行3回目、ループB実行1回目（繰り返し1-3）

同様に、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ106）。この時点で、フォーク箇所候補 f 0 および f 2 が選択済みなので、フォーク箇所候補 f 1 が選択される。フォーク箇所候補 f 1 はフォーク箇所候補の最良組合せ {f 0, f 2} に含まれていないので（ステップ107）、フォーク箇所候補の最良組合せ {f 0, f 2} にフォーク箇所候補 f 1 を加え、試行組合せは {f 0, f 1, f 2} となる（ステップ108）。次に、試行組合せの並列実行性能が評価されて、その結果である 3.5 が得られる（ステップ110）。試行組合せの並列実行性能（3.5）はフォーク箇所候補の最良組合せの並列実行性能（3.2）よりも良いので（ステップ111）、フォーク箇所候補の最良組合せは {f 0, f 1, f 2} となる（ステップ112）。ループAにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ113）、次にステップ106からループAの4回目の実行を開始する。

#### 【0096】

・ループA実行4回目、ループB実行1回目（繰り返し1-4）

同様に、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ106）。この時点で、フォーク箇所候補 f 0, f 1, f 2 が選択済みなので、フォーク箇所候補 f 3 が選択される。フォーク箇所候補 f 3 はフォーク箇所候補の最良組合せ {f 0, f 1, f 2} に含まれていないので（ステップ107）、フォーク箇所候補の最良組合せ {f 0, f 1, f 2} にフォーク箇所候補 f 3 を加え、試行組合せは {f 0, f 1, f 2, f 3} となる（ステップ108）。次に、試行組合せの並列実行性能が評価されて、その結果である 2.5 が得られる（ステップ110）。試行組合せの並列実行性能（2.5）はフォーク箇所候補の最良組合せの並列実行性能（3.5）よりも悪いので（ステップ111）、フォーク箇所候補の最良組合せは {f 0, f 1, f 2} のままである。ループAにおいてすべてのフォーク箇所候補を選択し（ステップ113）、またループA実行の3回目で新たにフォーク箇所候補の最良組合せが得られたので（ステップ114）、すべてのフォーク箇所候補

を未選択の状態にして（ステップ115）、次にステップ106からループB実行の2回目を開始する。

**【0097】**

・ループA実行1回目、ループB実行2回目（繰り返し2-1）

まず、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ106）。この時点では、すべてのフォーク箇所候補が未選択なので、フォーク箇所候補 f 0 が選択される。フォーク箇所候補 f 0 はフォーク箇所候補の最良組合せ {f 0, f 1, f 2} に含まれているので（ステップ107）、フォーク箇所候補の最良組合せ {f 0, f 1, f 2} からフォーク箇所候補 f 0 を取り除き、試行組合せは {f 1, f 2} となる（ステップ109）。次に、試行組合せの並列実行性能が評価されて、その結果である 3.8 が得られる（ステップ110）。試行組合せの並列実行性能（3.8）はフォーク箇所候補の最良組合せの並列実行性能（3.5）よりも良いので（ステップ111）、フォーク箇所候補の最良組合せは {f 1, f 2} となる（ステップ112）。ループAにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ113）、次にステップ106からループAの2回目の実行を開始する。

**【0098】**

以下、詳細な説明は省略するが、図5に示すような同様な動作を行う。

**【0099】**

・ループA実行2回目、ループB実行2回目（繰り返し2-2）

フォーク箇所候補 f 2 を選択して試行組合せは {f 1} となる。フォーク箇所候補の最良組合せは {f 1, f 2} のままである。

**【0100】**

・ループA実行3回目、ループB実行2回目（繰り返し2-3）

フォーク箇所候補 f 1 を選択して試行組合せは {f 2} となる。フォーク箇所候補の最良組合せは {f 1, f 2} のままである。

**【0101】**

・ループA実行4回目、ループB実行2回目（繰り返し2-4）

フォーク箇所候補  $f_3$  を選択して試行組合せは  $\{f_1, f_2, f_3\}$  となる。フォーク箇所候補の最良組合せは  $\{f_1, f_2\}$  のままである。ループAにおいてすべてのフォーク箇所候補を選択し（ステップ113）、またループAの実行の1回目で新たにフォーク箇所候補の最良組合せ  $\{f_1, f_2\}$  を得られたので（ステップ114）、すべてのフォーク箇所候補を未選択の状態にして（ステップ115）、次にステップ106からループB実行の3回目を開始する。

**【0102】**

・ループA実行1回目、ループB実行3回目（繰り返し3-1）

フォーク箇所候補  $f_0$  を選択して試行組合せは  $\{f_0, f_1, f_2\}$  となる。フォーク箇所候補の最良組合せは  $\{f_1, f_2\}$  のままである。

**【0103】**

・ループA実行2回目、ループB実行3回目（繰り返し3-2）

フォーク箇所候補  $f_2$  を選択して試行組合せは  $\{f_1\}$  となる。フォーク箇所候補の最良組合せは  $\{f_1, f_2\}$  のままである。

**【0104】**

・ループA実行3回目、ループB実行3回目（繰り返し3-3）

フォーク箇所候補  $f_1$  を選択して試行組合せは  $\{f_2\}$  となる。フォーク箇所候補の最良組合せは  $\{f_1, f_2\}$  のままである。

**【0105】**

・ループA実行4回目、ループB実行3回目（繰り返し3-4）

フォーク箇所候補  $f_3$  を選択して試行組合せは  $\{f_1, f_2, f_3\}$  となる。フォーク箇所候補の最良組合せは  $\{f_1, f_2\}$  のままである。ループAにおいてすべてのフォーク箇所候補を選択し（ステップ113）、またループAの繰り返し実行において新たにフォーク箇所候補の最良組合せが得られなかったので（ステップ114）、フォーク箇所候補の最良組合せである  $\{f_1, f_2\}$  をフォーク箇所候補最良組合せ決定処理の処理結果として出力し（ステップ116）、終了する。

**【0106】**

以上説明したように、第1の実施の形態の具体例によれば、フォーク箇所候補を



並列実行性能への効果があると予測される順序に順序付けし、その順序によって、その時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していく。フォーク箇所候補のすべての組合せについて並列実行性能を評価する方法では16回の評価が必要なのに比べて、第1の実施の形態による並列実行性能を評価する方法では12回の評価で済むことから明らかなように、フォーク箇所候補最良組合せ決定処理の処理時間を短縮することができるという効果がある。

#### 【0107】

なお、第1の実施の形態では、マルチプロセッサシステムのシミュレータ上で並列実行のシミュレーションを行って評価するようにしたが、マルチプロセッサシステム上で実際に実行して評価するようにしてもよい。詳しくは、並列実行性能を評価するフォーク箇所候補の試行組合せに対して、実際にフォーク命令を挿入して並列化プログラムを生成し、それを実際にマルチプロセッサシステム上で実行して並列実行性能を評価する。

#### 【0108】

##### [第2の実施の形態]

マルチスレッド実行方法としてフォーク1回モデルを採用した場合、スレッドはその生存中に高々1回に限って有効な子スレッドを生成することを保証する必要がある。これには、非特許文献1等で開示されているように、逐次処理プログラムから並列化プログラムを生成するコンパイルの段階で静的に制限する方法や、特許文献3で開示されているように、親スレッド実行中に選択することにより動的に保証する方法があるが、いずれの方式においても、フォーク1回モデル上で、同時に実行することのできないフォーク箇所候補の組合せが存在する。すなわち、フォーク1回モデルを仮定した場合、互いに同時に実行可能なフォーク箇所候補の試行組合せのみに対して並列実行性能の評価を行うことにすると、フォーク箇所候補最良組合せ決定処理の処理時間の短縮を期待することができる。

#### 【0109】

図6を参照すると、本発明の第2の実施の形態に係るプログラム並列化装置20'は、入力される逐次処理プログラム13の制御フローおよびデータフローの解

析を行う制御・データフロー解析部 21 と、制御・データフロー解析部 21 による制御フローおよびデータフローの解析結果を参照して逐次処理プログラム 13 のフォーク箇所候補を決定するフォーク箇所候補決定部 22 と、フォーク箇所候補決定部 22 により決定されたフォーク箇所候補からフォーク 1 回モデル上で互いに同時に実行可能なフォーク箇所候補の組合せのみからなる試行組合せを生成し、これを並列実行性能評価部 24 に与えて評価されたフォーク箇所候補の試行組合せの並列実行性能を基準にフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定部 23' と、フォーク箇所候補最良組合せ決定部 23' より与えられるフォーク箇所候補の試行組合せにより逐次処理プログラム 13 を並列化したときの並列実行性能を、ある入力データ 15 に対して評価する並列実行性能評価部 24 と、フォーク箇所候補最良組合せ決定部 23' により決定されたフォーク箇所候補の最良組合せに基づいて並列化プログラム 14 を生成し出力する並列化プログラム出力部 25 とを備える。

#### 【0110】

図 7 を参照すると、フォーク箇所候補決定部 22、フォーク箇所候補最良組合せ決定部 23'、および並列実行性能評価部 24 によるフォーク箇所候補最良組合せ決定処理は、フォーク箇所候補生成ステップ 120 と、フォーク箇所候補順序付けステップ 121 と、フォーク箇所候補初期組合せ生成ステップ 122 と、逐次実行トレース生成ステップ 123 と、スレッド要素分割ステップ 124 と、スレッド要素情報解析・記憶ステップ 125 と、フォーク箇所候補選択ステップ 126 と、フォーク箇所候補最良組合せ判定ステップ 127 と、フォーク箇所候補加入・試行組合せ設定ステップ 128 と、キャンセルフォーク箇所候補削除ステップ 137 と、フォーク箇所候補削除・試行組合せ設定ステップ 129 と、並列実行性能評価ステップ 130 と、並列実行性能比較ステップ 131 と、キャンセルフォーク箇所候補除去ステップ 138 と、最良組合せ設定ステップ 132 と、全フォーク箇所候補選択判定ステップ 133 と、新最良組合せ発見判定ステップ 134 と、全フォーク箇所候補未選択設定ステップ 135 と、フォーク箇所候補最良組合せ出力ステップ 136 とからなる。

#### 【0111】

図 7 のフォーク箇所候補最良組合せ決定処理は、フォーク 1 回制限の保証方法として、特許文献 3 に開示されている、親スレッドのフォーク命令毎に当該親スレッドから生成された子スレッドが既に存在する場合には、その子スレッドをキャンセルすることで、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する 1 つのフォーク命令を親スレッド実行中に選択する方法（特許文献 3 の請求項 2，段落 0 0 1 4 等参照）を採用すると仮定した場合の動作である。

#### 【0 1 1 2】

フォーク 1 回モデルの実現方法には、既述したように、主に逐次処理プログラムから並列化プログラムを生成するコンパイル段階で静的に制限する方法と、親スレッド実行中に選択することにより動的に保証する方法がある。後者の具体的な方法には、親スレッドのフォーク命令毎に当該親スレッドから生成された子スレッドがすでに存在する場合には、その子スレッドをキャンセルすることで、親スレッド中に存在する複数のフォーク命令のうちから有効な子スレッドを生成する 1 つのフォーク命令を親スレッド実行中に選択する方法があり、第 2 の実施の形態では、この方法を仮定する。

#### 【0 1 1 3】

この方法によると、フォーク箇所候補 f 0 とフォーク箇所候補 f 1 とを同時に選択した場合、フォーク箇所候補 f 1 は必ずフォーク箇所候補 f 0 によりキャンセルされると仮定すると、第 2 の実施の形態のフローチャートのループ A のある繰り返しにおいて、フォーク箇所候補 f 1 を選択し、それをフォーク箇所候補 f 0 を含む試行組合せに加える場合を考える。この場合、効果を評価したいフォーク箇所候補 f 1 は並列実行時にフォーク箇所候補 f 0 によりキャンセルされてしまう。ステップ 1 3 7 は、キャンセルの原因であるフォーク箇所候補 f 0 を試行組合せからあらかじめ取り除いて、フォーク箇所候補 f 1 が実行されるようにする処理に相当する。

#### 【0 1 1 4】

逆に、フローチャートのループ A のある繰り返しにおいてフォーク箇所候補 f 0 を選択し、それをフォーク箇所候補 f 1 を含む試行組合せに加える場合を考える

。この場合、効果を評価したいフォーク箇所候補 f 0 はキャンセルされないが、フォーク箇所候補 f 1 は必ずキャンセルされるため、フォーク箇所候補 f 1 が試行組合せに入っている意味がない。ステップ 138 は、この必ずキャンセルされて意味をなさないフォーク箇所候補をあらかじめ取り除く処理に相当する。

#### 【0115】

次に、このように構成された第 2 の実施の形態に係るプログラム並列化装置 20' の動作について説明する。

#### 【0116】

まず、制御・データフロー解析部 21 は、逐次処理プログラム 13 の制御フローおよびデータフローを解析する。

#### 【0117】

次に、フォーク箇所候補決定部 22 は、制御・データフロー解析部 21 による制御フローおよびデータフローの解析結果を参照して、フォーク箇所候補を生成する（ステップ 120）。

#### 【0118】

次に、フォーク箇所候補最良組合せ決定部 23' は、すべてのフォーク箇所候補のそれぞれの並列実行性能に対する効果を予測し、その効果順にフォーク箇所候補に順序を付ける（ステップ 121）。効果の予測は、例えば、逐次実行したときの逐次実行トレースの結果を参照して、予測対象のフォーク箇所候補のフォーク元からフォーク先までの実行サイクル数および該フォーク箇所候補の出現回数を参照して効果を予測してもよいし、フォーク箇所候補をそれぞれ単独で並列実行したときの並列実行性能を並列実行性能評価部 24 によって評価して効果を求めてもよい。さらに、フォーク 1 回制限を保証するためのフォークのキャンセルを発生させるフォーク箇所候補間の関係を求める。これは、逐次実行トレースの結果や制御フローグラフを参照して求めることができる。例えば、逐次処理プログラム 13 の実行フロー上で、あるフォーク箇所候補 f a のフォーク点からフォーク先までの間に、あるフォーク箇所候補 f b のフォーク点がある場合、実行時にフォーク箇所候補 f a はフォーク箇所候補 f b によってキャンセルされる。

#### 【0119】

続いて、フォーク箇所候補最良組合せ決定部 2 3' は、最初に並列実行性能を評価するフォーク箇所候補の初期組合せを生成し、それをフォーク箇所候補の最良組合せとする（ステップ 1 2 2）。これは、例えば、ステップ 1 2 1 で求められたフォーク箇所候補の順序で上位から一定数を含むフォーク箇所候補の組合せを初期組合せとし、その初期組合せの並列実行性能を並列実行性能評価部 2 4 によって評価し、初期組合せをこの時点でのフォーク箇所候補の最良組合せとする。

#### 【0 1 2 0】

並列実行性能評価部 2 4 は、まず、逐次処理プログラム 1 3 を入力データ 1 5 で逐次実行したときの逐次実行トレースを生成する（ステップ 1 2 3）。

#### 【0 1 2 1】

次に、並列実行性能評価部 2 4 は、逐次実行トレースをすべてのターム点候補を分割点として分割する（ステップ 1 2 4）。

#### 【0 1 2 2】

続いて、並列実行性能評価部 2 4 は、各スレッド要素毎の実行サイクル数、スレッド要素内に出現するフォーク箇所候補とその出現位置、フォーク先のスレッド要素、スレッド要素間のメモリ依存関係等のスレッド要素情報を解析し、それらを各スレッド要素毎に記憶する（ステップ 1 2 5）。

#### 【0 1 2 3】

次に、フォーク箇所候補最良組合せ決定部 2 3' は、ステップ 1 2 6 からステップ 1 3 3 までで構成されるループ A を繰り返し実行中に、未だ未選択のフォーク箇所候補の中で順序が一番上位のフォーク箇所候補を 1 つ選択する（ステップ 1 2 6）。

#### 【0 1 2 4】

続いて、フォーク箇所候補最良組合せ決定部 2 3' は、ステップ 1 2 6 で選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれているか否かを判定する（ステップ 1 2 7）。

#### 【0 1 2 5】

選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていなければ、フォーク箇所候補最良組合せ決定部 2 3' は、フォーク箇所候補の最良組

合せにステップ126で選択されたフォーク箇所候補を加え、そのフォーク箇所候補の組合せを試行組合せとする（ステップ128）。

【0126】

次に、フォーク箇所候補最良組合せ決定部23'は、ステップ126で選択されたフォーク箇所候補をキャンセルするフォーク箇所候補を試行組合せから取り除く（ステップ137）。

【0127】

一方、選択されたフォーク箇所候補がフォーク箇所候補の最良組合せに含まれていれば、フォーク箇所候補最良組合せ決定部23'は、フォーク箇所候補の最良組合せからステップ126で選択されたフォーク箇所候補を取り除き、そのフォーク箇所候補の組合せを試行組合せとする（ステップ129）。

【0128】

次に、フォーク箇所候補最良組合せ決定部23'は、試行組合せによる並列化の並列実行性能を、並列実行性能評価部24によって評価する（ステップ130）。

【0129】

続いて、フォーク箇所候補最良組合せ決定部23'は、試行組合せの並列実行性能と最良組合せの並列実行性能とを比較する（ステップ131）。試行組合せの並列実行性能の方が悪ければ、ステップ138およびステップ132をスキップする。

【0130】

試行組合せの並列実行性能の方が良ければ、フォーク箇所候補最良組合せ決定部23'は、ステップ126で選択されたフォーク箇所候補がキャンセルするフォーク箇所候補を試行組合せから取り除く（ステップ138）。

【0131】

次に、フォーク箇所候補最良組合せ決定部23'は、試行組合せを現時点でのフォーク箇所候補の最良組合せとする（ステップ132）。

【0132】

次に、フォーク箇所候補最良組合せ決定部23'は、ループAの繰り返し実行に

において、未選択のフォーク箇所候補があるかどうかを判定する（ステップ 1 3 3）。未選択のフォーク箇所候補があれば、ステップ 1 2 6 に制御を戻し、ループ A として繰り返し実行する。

#### 【0 1 3 3】

未選択のフォーク箇所候補がなければ、すなわちすべてのフォーク箇所候補を選択したならば、フォーク箇所候補最良組合せ決定部 2 3' は、直前のループ A の繰り返し実行において、新たにフォーク箇所候補の最良組合せが見つかったかどうかを判定する（ステップ 1 3 4）。

#### 【0 1 3 4】

新たにフォーク箇所候補の最良組合せが見つかった場合には、フォーク箇所候補最良組合せ決定部 2 3' は、ループ A の繰り返し実行に関して、すべてのフォーク箇所候補を未選択の状態にして（ステップ 1 3 5）、ステップ 1 2 6 に制御を戻し、ループ B として繰り返し実行する。

#### 【0 1 3 5】

新たにフォーク箇所候補の最良組合せが見つからなかった場合には、フォーク箇所候補最良組合せ決定部 2 3' は、決定されたフォーク箇所候補の最良組合せをフォーク箇所候補最良組合せ決定処理の結果として並列化プログラム出力部 2 5 に出力する（ステップ 1 3 6）。

#### 【0 1 3 6】

最後に、並列化プログラム出力部 2 5 は、フォーク箇所候補最良組合せ決定部 2 3' により決定された最良組合せの各フォーク箇所候補にフォーク命令を挿入して並列化プログラム 1 4 を生成し出力する。

#### 【0 1 3 7】

以上説明したように、第 2 の実施の形態によれば、マルチスレッド実行方法としてフォーク 1 回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを対象に並列実行性能を評価するので、フォーク箇所候補最良組合せ決定処理の処理時間を、第 1 の実施の形態に比べてさらに短縮することができるという効果が得られる。

#### 【0 1 3 8】

### [第 2 の実施の形態の具体例]

第 2 の実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を、図 4 および図 8 を参照して説明する。

#### 【0 1 3 9】

ここで、フォーク箇所候補として  $f_0$ ,  $f_1$ ,  $f_2$ ,  $f_3$  があるとする。このときのフォーク箇所候補の組合せの総数は、まったくフォークをせずに逐次実行した場合 ( $\phi$ ) を含めて 16 通り存在する。

#### 【0 1 4 0】

並列実行によって、逐次実行したとき ( $\phi$  の場合) の性能の何倍の性能が達成されるかを並列実行性能の指標として採用する。ここで、16 通りのフォーク箇所候補の組合せのそれぞれの並列実行性能を、図 4 に示すように仮定する。この場合、フォーク箇所候補として  $f_1$  および  $f_2$  を選択したときの並列実行性能が最も高く、逐次実行の 3.8 倍の性能である。

#### 【0 1 4 1】

次に、図 4 に示す具体例に対する第 2 の実施の形態によるフォーク箇所候補最良組合せ決定処理の動作の詳細を、図 7 のフローチャートも参照しつつ説明する。

#### 【0 1 4 2】

まず、ステップ 120 において、フォーク箇所候補として  $f_0$ ,  $f_1$ ,  $f_2$ ,  $f_3$  が得られているとする。

#### 【0 1 4 3】

次に、ステップ 121 において、フォーク箇所候補に順序を付ける。ここで、フォーク箇所候補それぞれを単独で並列実行したときの並列実行性能を指標として、それぞれの並列実行性能が大きい順序に順序付けを行うとする。図 4 に示すそれぞれの並列実行性能を参照すると、 $f_0$ ,  $f_2$ ,  $f_1$ ,  $f_3$  の順序になる。次に、フォーク箇所候補間のキャンセル関係を求める。ここでは、フォーク箇所候補  $f_0$  がフォーク箇所候補  $f_1$  をキャンセルし、フォーク箇所候補  $f_2$  がフォーク箇所候補  $f_3$  をキャンセルする関係が存在すると仮定する。

#### 【0 1 4 4】

続いて、ステップ 122 において、初期組合せを生成する。ステップ 121 で求



めたフォーク箇所候補の順序の上位から2つを初期組合せに採用すると、{f 0, f 2} が初期組合せになる。{f 0, f 2} の並列実行性能が評価され、その結果である3. 2が得られる。また、{f 0, f 2} は、この時点でのフォーク箇所候補の最良組合せになる。

#### 【0145】

次に、ステップ126からループAおよびループBを繰り返し実行される。各繰り返しにおける動作の様子を、図8に示す。

#### 【0146】

・ループA実行1回目、ループB実行1回目（繰り返し1-1）

まず、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ126）。この時点では、すべてのフォーク箇所候補が未選択なので、フォーク箇所候補f 0が選択される。フォーク箇所候補f 0はフォーク箇所候補の最良組合せ {f 0, f 2} に含まれているので（ステップ127）、フォーク箇所候補の最良組合せ {f 0, f 2} からフォーク箇所候補f 0を取り除き、試行組合せは {f 2} となる（ステップ129）。次に、試行組合せの並列実行性能が評価されて、その結果である2. 4が得られる（ステップ130）。試行組合せの並列実行性能（2. 4）はフォーク箇所候補の最良組合せの並列実行性能（3. 2）よりも悪いので（ステップ131）、フォーク箇所候補の最良組合せは {f 0, f 2} のままである。ループAにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ133）、次にステップ126からループAの2回目の実行を開始する。

#### 【0147】

・ループA実行2回目、ループB実行1回目（繰り返し1-2）

同様にして、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ126）。この時点で、フォーク箇所候補f 0が選択済みなので、フォーク箇所候補f 2が選択される。フォーク箇所候補f 2はフォーク箇所候補の最良組合せ {f 0, f 2} に含まれているので（ステップ127）、フォーク箇所候補の最良組合せ {f 0, f 2} からフォーク箇所候補f 2を取り除き、試行組合せは {f 0} となる（ステップ129）。

次に、試行組合せの並列実行性能が評価されて、その結果である 2. 5 が得られる (ステップ 130)。試行組合せの並列実行性能 (2. 5) はフォーク箇所候補の最良組合せの並列実行性能 (3. 2) よりも悪いので (ステップ 131)、フォーク箇所候補の最良組合せは {f 0, f 2} のままである。ループ A において未だ未選択のフォーク箇所候補が残っているので (ステップ 133)、次にステップ 126 からループ A の 3 回目の実行を開始する。

#### 【0148】

・ループ A 実行 3 回目、ループ B 実行 1 回目 (繰り返し 1-3)

同様に、ループ A の実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する (ステップ 126)。この時点で、フォーク箇所候補 f 0 および f 2 が選択済みなので、フォーク箇所候補 f 1 が選択される。フォーク箇所候補 f 1 はフォーク箇所候補の最良組合せ {f 0, f 2} に含まれていないので (ステップ 127)、フォーク箇所候補の最良組合せ {f 0, f 2} にフォーク箇所候補 f 1 を加え、試行組合せは {f 0, f 1, f 2} となる (ステップ 128)。次に、選択されたフォーク箇所候補 f 1 がフォーク箇所候補 f 0 にキャンセルされるので、試行組合せ {f 0, f 1, f 2} からフォーク箇所候補 f 0 は取り除かれて、試行組合せが {f 1, f 2} となる (ステップ 137)。次に、試行組合せの並列実行性能が評価されて、その結果である 3. 8 が得られる (ステップ 130)。試行組合せの並列実行性能 (3. 8) はフォーク箇所候補の最良組合せの並列実行性能 (3. 2) よりも良いので (ステップ 131)、フォーク箇所候補の最良組合せは {f 1, f 2} となる (ステップ 132)。ループ A において未だ未選択のフォーク箇所候補が残っているので (ステップ 133)、次にステップ 126 からループ A の 4 回目の実行を開始する。

#### 【0149】

・ループ A 実行 4 回目、ループ B 実行 1 回目 (繰り返し 1-4)

同様に、ループ A の実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する (ステップ 126)。この時点で、フォーク箇所候補 f 0, f 1, f 2 が選択済みなので、フォーク箇所候補 f 3 が選択される。フォーク箇所候補 f 3 はフォーク箇所候補の最良組合せ {f 1, f 2} に含ま

れていないので（ステップ127）、フォーク箇所候補の最良組合せ  $\{f1, f2\}$  にフォーク箇所候補  $f3$  を加え、試行組合せは  $\{f1, f2, f3\}$  となる（ステップ128）。次に、選択されたフォーク箇所候補  $f3$  がフォーク箇所候補  $f2$  にキャンセルされるので、試行組合せ  $\{f1, f2, f3\}$  からフォーク箇所候補  $f2$  は取り除かれて、試行組合せが  $\{f1, f3\}$  となる（ステップ137）。次に、試行組合せの並列実行性能が評価されて、その結果である2.8が得られる（ステップ130）。試行組合せの並列実行性能（2.8）はフォーク箇所候補の最良組合せ  $\{f1, f2\}$  の並列実行性能（3.8）よりも悪いので（ステップ131）、フォーク箇所候補の最良組合せは  $\{f1, f2\}$  のままである。ループAにおいてすべてのフォーク箇所候補を選択し（ステップ133）、またループAの実行の3回目で新たにフォーク箇所候補の最良組合せを得られたので（ステップ134）、すべてのフォーク箇所候補を未選択の状態にして（ステップ135）、次にステップ126からループB実行の2回目を開始する。

#### 【0150】

・ループA実行1回目、ループB実行2回目（繰り返し2-1）

まず、ループAの実行において未選択のフォーク箇所候補で順序が一番上位のフォーク箇所候補を選択する（ステップ126）。この時点ではすべてのフォーク箇所候補が未選択なので、フォーク箇所候補  $f0$  が選択される。フォーク箇所候補  $f0$  はフォーク箇所候補の最良組合せ  $\{f1, f2\}$  に含まれていないので（ステップ127）、フォーク箇所候補の最良組合せ  $\{f1, f2\}$  にフォーク箇所候補  $f0$  を加え、試行組合せは  $\{f0, f1, f2\}$  となる（ステップ128）。次に、試行組合せの並列実行性能が評価されて、その結果である3.5が得られる（ステップ130）。試行組合せの並列実行性能（3.5）はフォーク箇所候補の最良組合せ  $\{f1, f2\}$  の並列実行性能（3.8）よりも悪いので（ステップ131）、フォーク箇所候補の最良組合せは  $\{f1, f2\}$  のままである。ループAにおいて未だ未選択のフォーク箇所候補が残っているので（ステップ133）、次にステップ126からループAの2回目の実行を開始する。

#### 【0151】

以下、詳細な説明は省略するが、図 8 に示すような同様な動作を行う。

【0 1 5 2】

・ ループ A 実行 2 回目、ループ B 実行 2 回目（繰り返し 2 - 2）

フォーク箇所候補 f 2 を選択して試行組合せは {f 1} となる。フォーク箇所候補の最良組合せは {f 1, f 2} のままである。

【0 1 5 3】

・ ループ A 実行 3 回目、ループ B 実行 2 回目（繰り返し 2 - 3）

フォーク箇所候補 f 1 を選択して試行組合せは {f 2} となる。フォーク箇所候補の最良組合せは {f 1, f 2} のままである。

【0 1 5 4】

・ ループ A 実行 4 回目、ループ B 実行 2 回目（繰り返し 2 - 4）

フォーク箇所候補 f 3 を選択して試行組合せは {f 1, f 3} となる。フォーク箇所候補の最良組合せは {f 1, f 2} のままである。ループ A においてすべてのフォーク箇所候補を選択し（ステップ 1 3 3）、またループ A の繰り返し実行において新たにフォーク箇所候補の最良組合せが得られなかったので（ステップ 1 3 4）、フォーク箇所候補の最良組合せである {f 1, f 2} をフォーク箇所候補最良組合せ決定処理の処理結果として出力し（ステップ 1 3 6）、終了する。

【0 1 5 5】

以上説明したように第 2 の実施の形態の具体例によれば、マルチスレッド実行方法としてフォーク 1 回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを対象に並列実行性能を評価するので、8 回の評価でフォーク箇所候補最良組合せ決定処理が終了することから明らかなように、第 1 の実施の形態の具体例に比べて、さらにフォーク箇所候補最良組合せ決定処理の処理時間を短縮できるという効果が得られる。

【0 1 5 6】

[第 3 の実施の形態]

フォーク箇所候補の組合せの総数はフォーク箇所候補の数に対して指数関数的に増加するため、通常、フォーク箇所候補の数が増加するにつれてフォーク箇所候

補最良組合せ決定処理の処理時間も急速に増加してしまう。これに対し、フォーク箇所候補を適当数にグループ分割し、各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うと、フォーク箇所候補の組合せの総数を削減することができるため、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができる。このとき、並列実行性能のより高いフォーク箇所候補を得るためにはフォーク箇所候補のグループ分割が特に重要であり、独立なフォーク箇所候補最良組合せ決定処理を可能にするために、グループ間ではなるべく互いに影響を及ぼし合わないようグループ分割することが必要である。第3の実施の形態では、フォーク箇所候補間のキャンセル関係を参照し、キャンセル関係があれば関係が強く、なければ弱いとしてグループ分割することでこれを達成する。

#### 【0157】

図9を参照すると、本発明の第3の実施の形態に係るプログラム並列化装置20”は、入力される逐次処理プログラム13の制御フローおよびデータフローの解析を行う制御・データフロー解析部21と、制御・データフロー解析部21による制御フローおよびデータフローの解析結果を参照して逐次処理プログラム13のフォーク箇所候補を決定するフォーク箇所候補決定部22と、フォーク箇所候補決定部22により決定された全フォーク箇所候補の集合を互いになるべく影響を及ぼし合わないようフォーク箇所候補のグループに分割し、分割されたフォーク箇所候補のグループの中でフォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補の試行組合せを生成し、それによる並列実行性能を並列実行性能評価部24によって評価した結果を参照してフォーク箇所候補の最良組合せを決定するフォーク箇所候補最良組合せ決定処理を行い、各グループ毎の処理結果であるフォーク箇所候補の最良組合せの和を全体の処理結果として決定するフォーク箇所候補最良組合せ決定部23”と、フォーク箇所候補最良組合せ決定部23”より与えられるフォーク箇所候補の試行組合せにより逐次処理プログラム13を並列化したときの並列実行性能を、ある入力データ15に対して評価する並列実行性能評価部24と、フォーク箇所候補最良組合せ決定部23”により決定されたフォーク箇所候補の試行組合せに基づいて並列

化プログラム 14 を生成し出力する並列化プログラム出力部 25 とを備える。

【0158】

図 10 (a) を参照すると、フォーク箇所候補最良組合せ決定部 23” によるフォーク箇所候補最良組合せ決定処理は、グループ分割処理呼び出しステップ 140 と、グループに対するフォーク箇所候補最良組合せ決定処理ステップ 141 と、全グループ処理判定ステップ 142 と、結果出力ステップ 143 とからなる。

【0159】

図 10 (b) を参照すると、フォーク箇所候補最良組合せ決定部 23” によるフォーク箇所候補のグループ分割処理は、フォーク箇所候補数・分割下限数比較ステップ 146 と、フォーク箇所候補分割ステップ 147 と、集合のグループへの分割ステップ 148 と、グループ分割処理呼び出しステップ 149 と、グループ分割処理呼び出しステップ 150 とからなる。

【0160】

次に、このように構成された第 3 の実施の形態に係るプログラム並列化装置 20” の動作について説明する。

【0161】

制御・データフロー解析部 21 およびフォーク箇所候補決定部 22 の処理は、第 1 および第 2 の実施の形態における動作と全く同様であるので、説明を省略する。

【0162】

フォーク箇所候補決定部 22 の処理が終了すると、フォーク箇所候補最良組合せ決定部 23” は、フォーク箇所候補決定部 22 により決定された全フォーク箇所候補の集合を F として、図 4 (b) のフローチャートに示すフォーク箇所候補のグループ分割処理を呼び出す (ステップ 140)。

【0163】

フォーク箇所候補のグループ分割処理では、フォーク箇所候補最良組合せ決定部 23” は、フォーク箇所候補間のキャンセル関係を参照して 2 分木状にフォーク箇所候補をグループに分割するものとする。なお、分割下限数 M および指定数 N は適切に与えられる数であり、例えば  $M=300$ 、 $N=M/2$  とすることができ

る。

#### 【0164】

ステップ140から同図(b)に示すフォーク箇所候補のグループ分割処理が呼び出されると、フォーク箇所候補最良組合せ決定部23”は、まず、与えられたフォーク箇所候補の集合Fに属するフォーク箇所候補の数が指定された分割下限数Mより大きければ(ステップ146)、ステップ147から集合Fのグループ分割処理を開始する。小さければ、フォーク箇所候補最良組合せ決定部23”は、グループ分割処理を行わずにフォーク箇所候補のグループ分割処理を呼び出した元(ステップ140、ステップ149またはステップ150)にリターンする。

#### 【0165】

次に、フォーク箇所候補最良組合せ決定部23”は、自身をキャンセルするフォーク箇所候補の数が指定数Nより大きいフォーク箇所候補の集合を集合Fから分割して新たにグループを生成する(ステップ147)。

#### 【0166】

続いて、フォーク箇所候補最良組合せ決定部23”は、集合Fをさらに2つのグループFaおよびFbにグループ分割する(ステップ148)。このとき、グループFaに属するフォーク箇所候補をキャンセルするフォーク箇所候補の中でグループFbに属するフォーク箇所候補の数、およびグループFbに属するフォーク箇所候補をキャンセルするフォーク箇所候補の中でグループFaに属するフォーク箇所候補の数の総和をなるべく少なくするようにグループ分割を行う。これは、互いになるべくフォーク箇所候補間で相関がないようにグループ分割するためである。このとき、2つのグループFaおよびFbにそれぞれ属するフォーク箇所候補の数は同等の数である必要はなく、上記総和が小さくなる限り差があってもよい。

#### 【0167】

次に、フォーク箇所候補最良組合せ決定部23”は、グループFaを集合Fとしてフォーク箇所候補のグループ分割処理を再帰的に呼び出してグループFaのグループ分割を行う(ステップ149)。

**【0168】**

同様に、フォーク箇所候補最良組合せ決定部 23” は、グループ F b を集合 F としてフォーク箇所候補のグループ分割処理を再帰的に呼び出してグループ F b のグループ分割を行う（ステップ 150）。

**【0169】**

この後、フォーク箇所候補最良組合せ決定部 23” は、フォーク箇所候補のグループ分割処理を呼び出した元（ステップ 140，ステップ 149 またはステップ 150）にリターンする。

**【0170】**

次に、フォーク箇所候補最良組合せ決定部 23” は、ステップ 140 で分割されたフォーク箇所候補のグループの中で、フォーク箇所候補最良組合せ決定処理を未だ行っていないグループに対してフォーク箇所候補最良組合せ決定処理を行う（ステップ 141）。このフォーク箇所候補最良組合せ決定処理は、第 1 の実施の形態あるいは第 2 の実施の形態におけるフォーク箇所候補最良組合せ決定処理を適用してもよいし、各グループ毎のフォーク箇所候補の数が十分少ない場合は、グループ内のフォーク箇所候補のすべての組合せについて並列実行性能の評価を行ってフォーク箇所候補の最良組合せを決定してもよい。第 1 の実施の形態または第 2 の実施の形態におけるフォーク箇所候補最良組合せ決定処理を適用する場合、フォーク箇所候補の初期組合せをグループ内のフォーク箇所候補のみから構成することもできる。

**【0171】**

続いて、フォーク箇所候補最良組合せ決定部 23” は、全グループの処理が完了したかどうかを判定し（ステップ 142）、未処理のグループがあるならば、ステップ 141 に制御を戻して、未処理のグループに対するフォーク箇所候補最良組合せ決定処理を繰り返す。

**【0172】**

全グループの処理が完了した場合は、フォーク箇所候補最良組合せ決定部 23” は、各グループ毎の処理結果であるフォーク箇所候補の組合せの和を全体の結果として出力する（ステップ 143）。



**【0173】****[第3の実施の形態の具体例]**

図11を参照して、第3の実施の形態に係るプログラム並列化装置20”におけるフォーク箇所候補のグループ分割処理の具体例を説明する。

**【0174】**

全フォーク箇所候補の集合を集合Fとし、集合Fのグループ分割を行うとする。集合Fのフォーク箇所候補の数が分割下限数M以上とすると（ステップ146）、まず集合Fから自身をキャンセルするフォーク箇所候補の数が指定数N以上のフォーク箇所候補を分離して、新たなグループF0とする（ステップ147）。次に、集合Fを2つのグループF0-aとF0-bとに分割する（ステップ148）。すなわち、集合Fは、3つのフォーク箇所候補のグループF0、F0-a、F0-bに分割される。次に、グループF0-aおよびグループF0-bに対して再びフォーク箇所候補のグループ分割処理を行う（ステップ149および150）。

**【0175】**

同様に、グループF0-aのフォーク箇所候補の数が分割下限数M以上とすると、グループF0-aは、グループF1、F1-a、F1-bに分割される。グループF1-aおよびF1-bに対して再びフォーク箇所候補のグループ分割処理を行う。

**【0176】**

同様に、グループF1-aのフォーク箇所候補の数が分割下限数M以上とすると、グループF1-aは、グループF2、F2-a、F2-bに分割される。グループF2-aおよびF2-bに対して再びフォーク箇所候補のグループ分割処理を行う。

**【0177】**

次に、グループF2-aのフォーク箇所候補の数が分割下限数M以下とすると、これ以上分割されずグループF2-aをグループF3とする。

**【0178】**

次に、グループF2-bのフォーク箇所候補の数が分割下限数M以下とすると、

これ以上分割されずグループF 2 - bをグループF 4 とする。

【0179】

同様に、グループF 1 - bのフォーク箇所候補の数が分割下限数M以下とすると、これ以上分割されずグループF 1 - bをグループF 5 とする。

【0180】

次に、グループF 0 - bのフォーク箇所候補の数が分割下限数M以上とすると、グループF 0 - bは、グループF 6, F 6 - a, F 6 - bに分割される。グループF 6 - aおよびグループF 6 - bに対して再びフォーク箇所候補のグループ分割処理を行う。

【0181】

次に、グループF 6 - aのフォーク箇所候補の数が分割下限数M以下とすると、これ以上分割されずグループF 6 - aをグループF 7 とする。

【0182】

次に、グループF 6 - bのフォーク箇所候補の数が分割下限数M以下とすると、これ以上分割されずグループF 6 - bをグループF 8 とする。

【0183】

以上により、全フォーク箇所候補の集合F aが、相互に及ぼす影響の少ないフォーク箇所候補のグループF 0 ~ F 8 に分割された。

【0184】

以上に説明したように、第3の実施の形態によれば、フォーク箇所候補をグループ間では互いになるべく影響を及ぼし合わないよう適当数のグループに分割して、各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うので、フォーク箇所候補の組合せの総数を削減することができ、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができる。

【0185】

[第4の実施の形態]

図12は、本発明の第4の実施の形態に係るプログラム並列化装置20の構成を示すブロック図である。本実施の形態に係るプログラム並列化装置20は、図1に示した第1の実施の形態に係るプログラム並列化装置20に対してプログラム

並列化プログラム 4 0 0 を付加するようにした点だけが異なる。したがって、その他の特に言及しない部分には同一符号を付して、それらの詳しい説明を省略する。

#### 【0 1 8 6】

プログラム並列化プログラム 4 0 0 は、コンピュータでなるプログラム並列化装置 2 0 に読み込まれ、プログラム並列化装置 2 0 の動作を、制御・データフロー解析部 2 1，フォーク箇所候補決定部 2 2，フォーク箇所候補最良組合せ決定部 2 3，並列実行性能評価部 2 4，および並列化プログラム出力部 2 5 として制御する。プログラム並列化プログラム 4 0 0 の制御によるプログラム並列化装置 2 0 の動作は、第 1 の実施の形態におけるプログラム並列化装置 2 0 の動作と全く同様になるので、その詳しい説明を割愛する。

#### 【0 1 8 7】

##### [第 5 の実施の形態]

図 1 3 は、本発明の第 5 の実施の形態に係るプログラム並列化装置 2 0' の構成を示すブロック図である。本実施の形態に係るプログラム並列化装置 2 0' は、図 6 に示した第 2 の実施の形態に係るプログラム並列化装置 2 0' に対してプログラム並列化プログラム 4 0 0' を付加するようにした点だけが異なる。したがって、その他の特に言及しない部分には同一符号を付して、それらの詳しい説明を省略する。

#### 【0 1 8 8】

プログラム並列化プログラム 4 0 0' は、コンピュータでなるプログラム並列化装置 2 0' に読み込まれ、プログラム並列化装置 2 0' の動作を、制御・データフロー解析部 2 1，フォーク箇所候補決定部 2 2，フォーク箇所候補最良組合せ決定部 2 3'，並列実行性能評価部 2 4，および並列化プログラム出力部 2 5 として制御する。プログラム並列化プログラム 4 0 0' の制御によるプログラム並列化装置 2 0' の動作は、第 2 の実施の形態におけるプログラム並列化装置 2 0' の動作と全く同様になるので、その詳しい説明を割愛する。

#### 【0 1 8 9】

##### [第 6 の実施の形態]

図 1 4 は、本発明の第 6 の実施の形態に係るプログラム並列化装置 2 0” の構成を示すブロック図である。本実施の形態に係るプログラム並列化装置 2 0” は、図 9 に示した第 3 の実施の形態に係るプログラム並列化装置 2 0” に対してプログラム並列化プログラム 4 0 0” を付加するようにした点だけが異なる。したがって、その他の特に言及しない部分には同一符号を付して、それらの詳しい説明を省略する。

#### 【0 1 9 0】

プログラム並列化プログラム 4 0 0” は、コンピュータでなるプログラム並列化装置 2 0” に読み込まれ、プログラム並列化装置 2 0” の動作を、制御・データフロー解析部 2 1，フォーク箇所候補決定部 2 2，フォーク箇所候補最良組合せ決定部 2 3”，並列実行性能評価部 2 4，および並列化プログラム出力部 2 5 として制御する。プログラム並列化プログラム 4 0 0” の制御によるプログラム並列化装置 2 0” の動作は、第 3 の実施の形態におけるプログラム並列化装置 2 0” の動作と全く同様になるので、その詳しい説明を割愛する。

#### 【0 1 9 1】

##### 【発明の効果】

第 1 の効果は、フォーク命令を逐次処理プログラムにより適切に挿入することができ、並列実行性能のより高い並列化プログラムを得ることができることである。その理由は、フォーク箇所候補の試行組合せに対して並列実行性能を直接評価した結果を基準にフォーク箇所候補の最良組合せを決定するからである。

#### 【0 1 9 2】

第 2 の効果は、フォーク箇所候補のすべての組合せに対して並列実行性能を評価する場合に比べて、フォーク箇所候補最良組合せ決定処理の処理時間を大幅に短縮できるということである。その理由は、フォーク箇所候補を並列実行性能への効果があると予測される順序に順序付けし、その順序に従ってその時点のフォーク箇所候補の最良組合せを基準に並列実行性能を評価してより良い組合せを構成していくためである。

#### 【0 1 9 3】

第 3 の効果は、フォーク箇所候補最良組合せ決定処理の処理時間をさらに短縮で

きることである。その理由は、マルチスレッド実行方法としてフォーク 1 回モデルを採用した場合に、互いに同時に実行可能なフォーク箇所候補の試行組合せのみを対象に並列実行性能を評価するからである。

#### 【0194】

第 4 の効果は、フォーク箇所候補の組合せの総数を削減することができ、フォーク箇所候補の数が多い場合でもフォーク箇所候補最良組合せ決定処理を高速に行うことができることである。その理由は、フォーク箇所候補をグループ間では互いになるべく影響を及ぼし合わないよう適当数のグループに分割して各グループ毎に独立にフォーク箇所候補最良組合せ決定処理を行うからである。

#### 【0195】

第 5 の効果は、並列実行性能を高速に評価することが可能であり、従ってフォーク箇所候補最良組合せ決定処理を高速に行うことができることである。その理由は、ターム点候補で決定されるスレッド要素の単位で並列実行のシミュレーションを行うようにしたからである。

#### 【図面の簡単な説明】

##### 【図 1】

本発明の第 1 の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

##### 【図 2】

本発明の第 1 の実施の形態におけるフォーク箇所候補最良組合せ決定処理を示すフローチャートである。

##### 【図 3】

フォーク 1 回モデルにおけるマルチスレッド実行方法の一例を示す図である。

##### 【図 4】

第 1 の実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を説明する図である。

##### 【図 5】

第 1 の本実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を説明する図である。

**【図 6】**

本発明の第 2 の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

**【図 7】**

本発明の第 2 の実施の形態におけるフォーク箇所候補最良組合せ決定処理を示すフローチャートである。

**【図 8】**

第 2 の実施の形態によるフォーク箇所候補最良組合せ決定処理の具体例を説明する図である。

**【図 9】**

本発明の第 3 の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

**【図 10】**

本発明の第 3 の実施の形態におけるフォーク箇所候補最良組合せ決定処理を示すフローチャートである。

**【図 11】**

本発明の第 3 の実施の形態におけるフォーク箇所候補のグループ分割の一例を示す図である。

**【図 12】**

本発明の第 4 の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

**【図 13】**

本発明の第 5 の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

**【図 14】**

本発明の第 6 の実施の形態に係るプログラム並列化装置の構成を示すブロック図である。

**【図 15】**

マルチスレッド実行方法の概要を説明する図である。

## 【図 1 6】

従来のプログラム並列化装置の構成例を示すブロック図である。

## 【符号の説明】

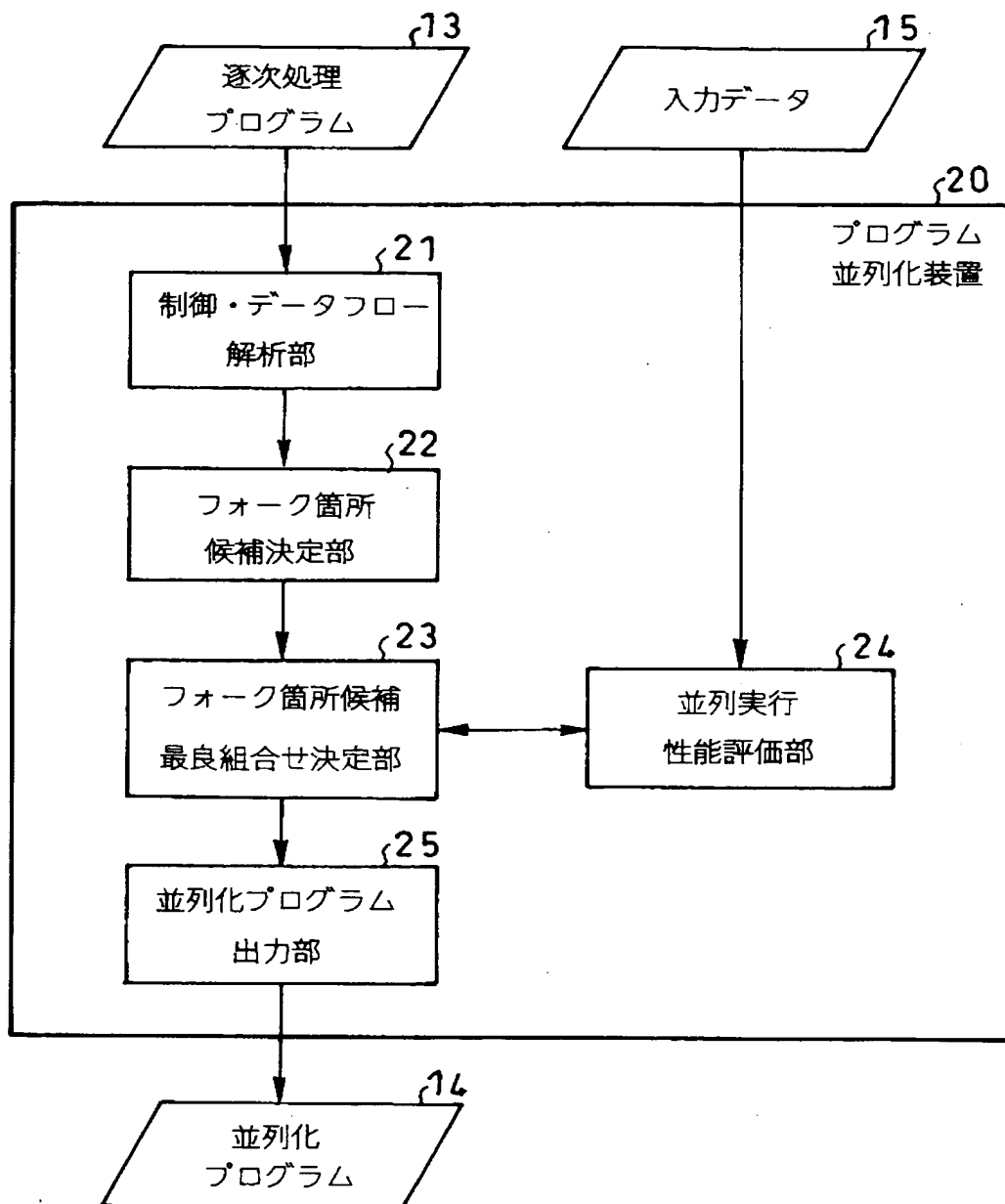
- 1 3 逐次処理プログラム
- 1 4 並列化プログラム
- 1 5 入力データ
- 2 0, 2 0', 2 0'' プログラム並列化装置
- 2 1 制御・データフロー解析部
- 2 2 フォーク箇所候補決定部
- 2 3, 2 3', 2 3'' フォーク箇所候補最良組合せ決定部
- 2 4 並列実行性能評価部
- 2 5 並列化プログラム出力部
- 1 0 0, 1 2 0 フォーク箇所候補生成ステップ
- 1 0 1, 1 2 1 フォーク箇所候補順序付けステップ
- 1 0 2, 1 2 2 フォーク箇所候補初期組合せ生成ステップ
- 1 0 3, 1 2 3 逐次実行トレース生成ステップ
- 1 0 4, 1 2 4 スレッド要素分割ステップ
- 1 0 5, 1 2 5 スレッド要素情報解析・記憶ステップ
- 1 0 6, 1 2 6 フォーク箇所候補選択ステップ
- 1 0 7, 1 2 7 フォーク箇所候補最良組合せ判定ステップ
- 1 0 8, 1 2 8 フォーク箇所候補加入・試行組合せ設定ステップ
- 1 0 9, 1 2 9 フォーク箇所候補削除・試行組合せ設定ステップ
- 1 1 0, 1 3 0 並列実行性能評価ステップ
- 1 1 1, 1 3 1 並列実行性能比較ステップ
- 1 1 2, 1 3 2 最良組合せ設定ステップ
- 1 1 3, 1 3 3 全フォーク箇所候補選択判定ステップ
- 1 1 4, 1 3 4 新最良組合せ発見判定ステップ
- 1 1 5, 1 3 5 全フォーク箇所候補未選択設定ステップ
- 1 1 6, 1 3 6 最良組合せ出力ステップ

- 1 3 7 キャンセルフォーク箇所候補削除ステップ
- 1 3 8 キャンセルフォーク箇所候補削除ステップ
- 1 4 0 グループ分割処理呼び出しステップ
- 1 4 1 グループに対するフォーク箇所候補最良組合せ決定処理ステップ
- 1 4 2 全グループ処理判定ステップ
- 1 4 3 結果出力ステップ
- 1 4 6 フォーク箇所候補数・分割下限数比較ステップ
- 1 4 7 フォーク箇所候補分割ステップ
- 1 4 8 集合のグループへの分割ステップ
- 1 4 9 グループ分割処理呼び出しステップ
- 1 5 0 グループ分割処理呼び出しステップ
- 4 0 0 , 4 0 0 ' , 4 0 0 " プログラム並列化プログラム

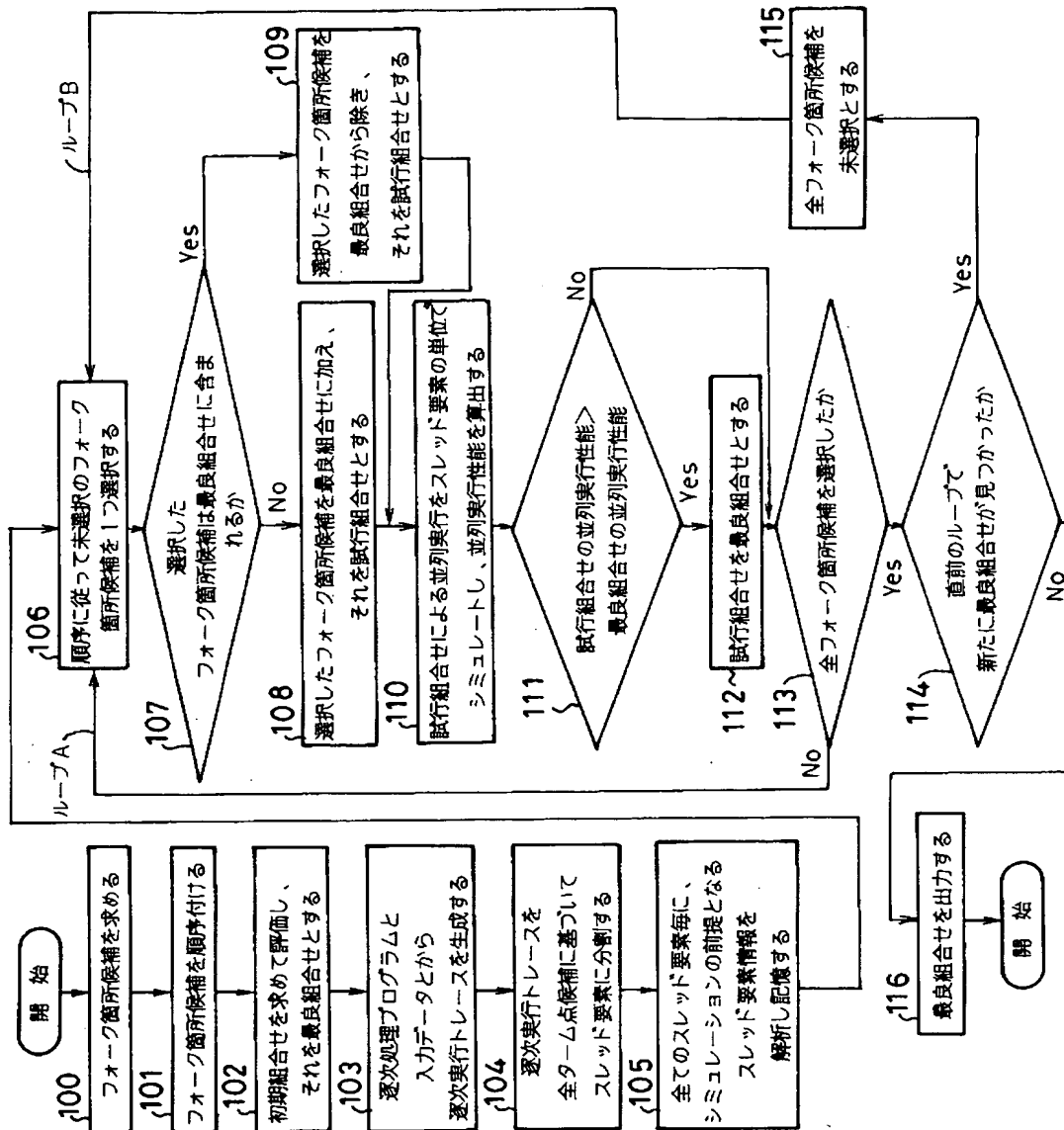


【書類名】 図面

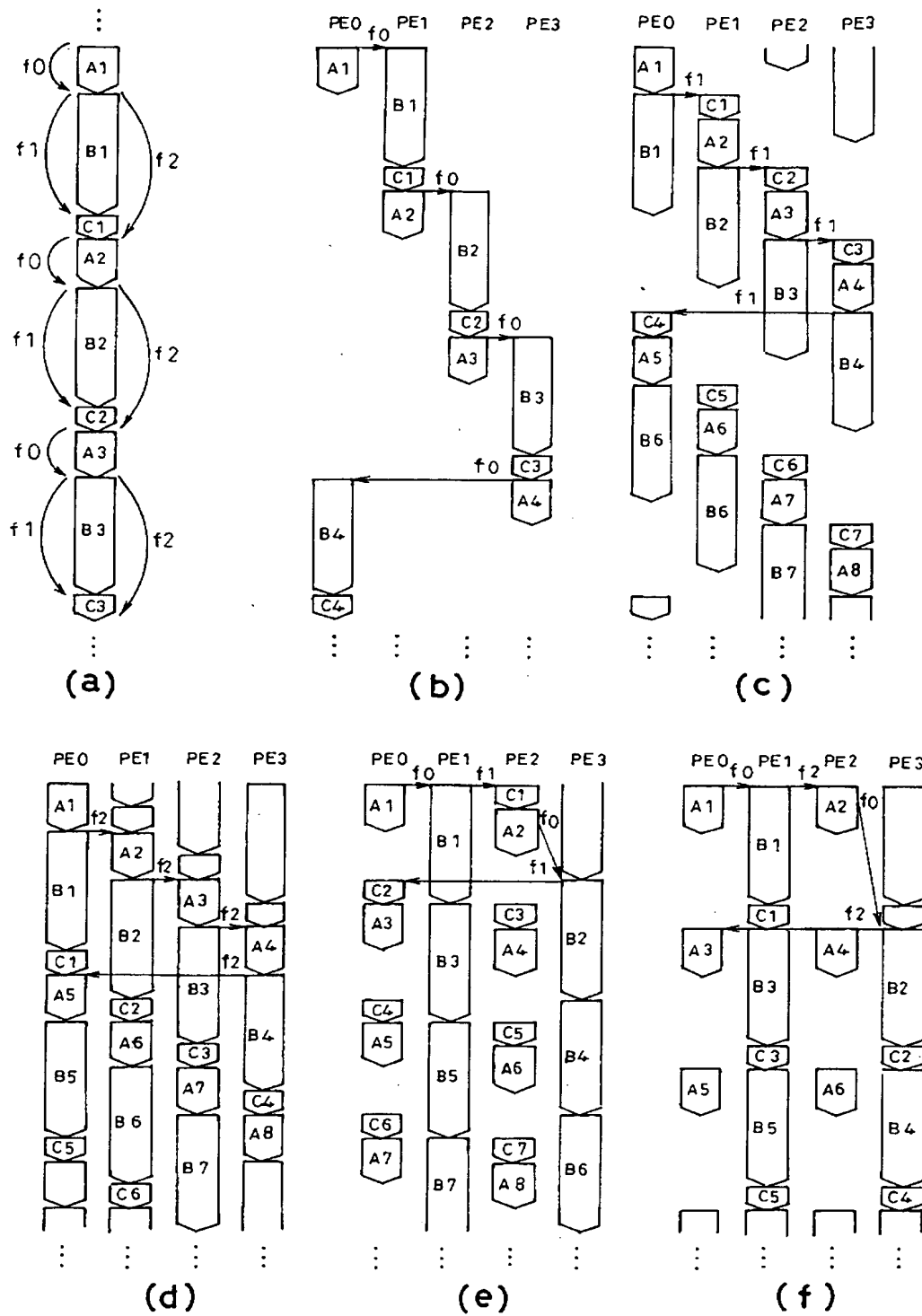
【図 1】



【図 2】



【図 3】



【図 4】

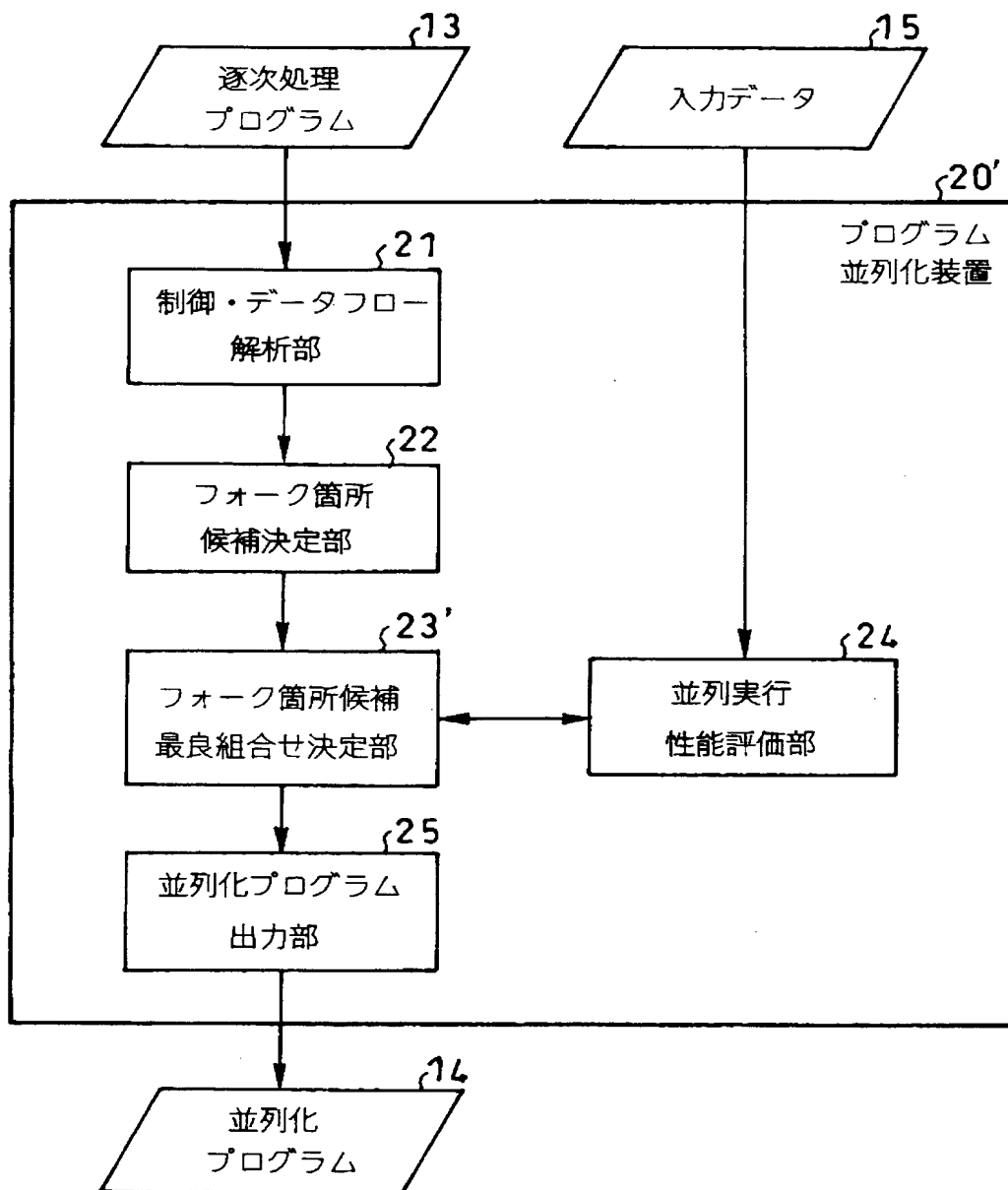
フォーク 箇所候補組合せ	並列実行性能(倍)
$\phi$	1.0
{f0}	2.5
{f1}	2.2
{f2}	2.4
{f3}	1.8
{f0, f1}	3.0
{f0, f2}	3.2
{f0, f3}	2.4
{f1, f2}	3.8
{f1, f3}	2.8
{f2, f3}	2.0
{f0, f1, f2}	3.5
{f0, f1, f3}	2.8
{f0, f2, f3}	2.5
{f1, f2, f3}	2.6
{f0, f1, f2, f3}	2.5

【図 5】

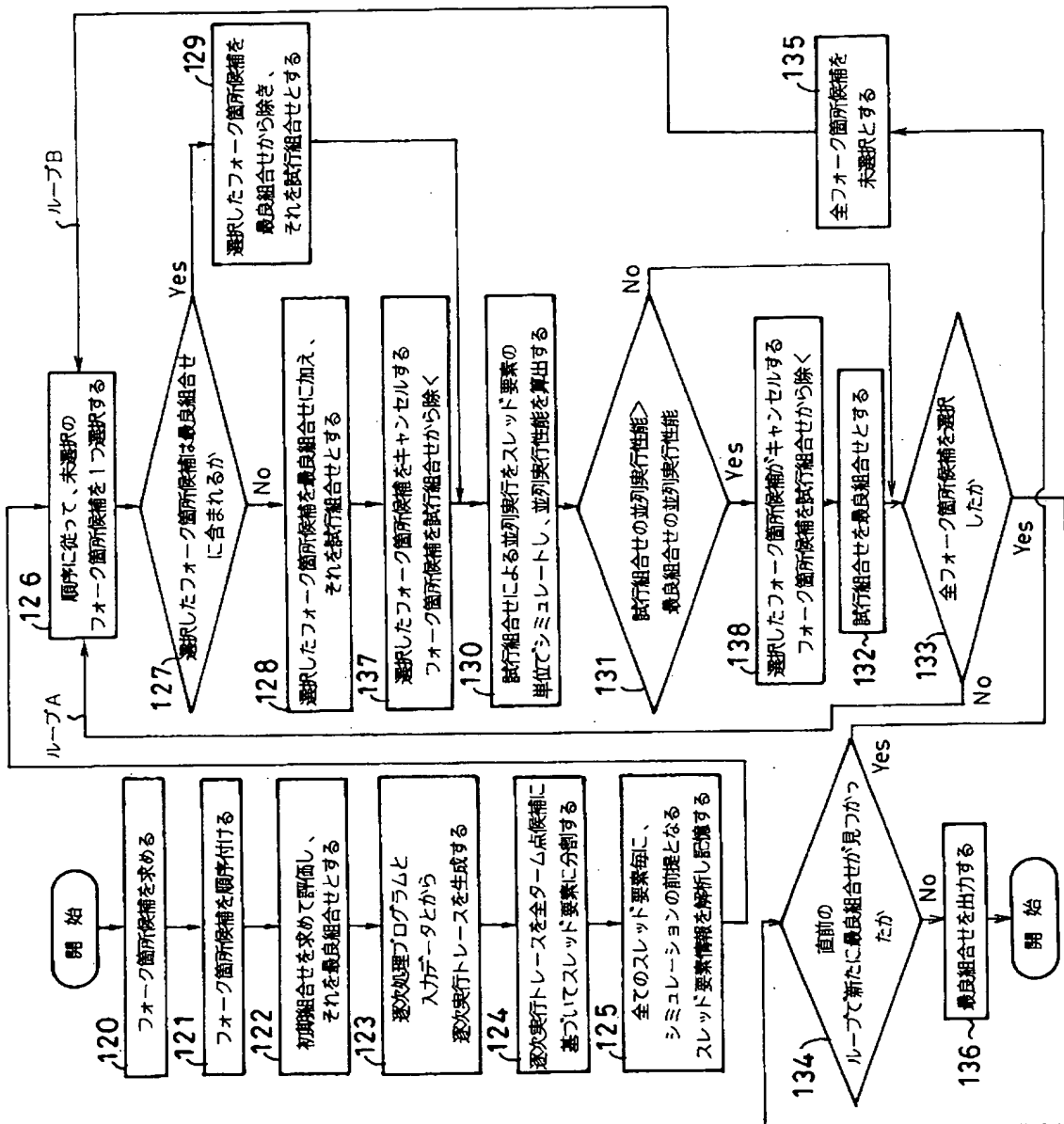
初期組合せ: {f0, f2} (並列実行性能: 3.2倍)

選択フオーク箇所候補		並列実行性能		選択済フオーク箇所候補	
	試行組合せ		最良組合せ		
繰り返し1-1	f0 {f2}	2.4	{f0, f2}	f0	
繰り返し1-2	f2 {f0}	2.5	{f0, f2}	f0, f2	
繰り返し1-3	f1 {f0, f1, f2}	3.5	{f0, f1, f2}	f0, f1, f2	
繰り返し1-4	f3 {f0, f1, f2, f3}	2.5	{f0, f1, f2}	f0, f1, f2, f3	
繰り返し2-1	f0 {f1, f2}	3.8	{f0, f1, f2}	f0	
繰り返し2-2	f2 {f1}	2.2	{f1, f2}	f0, f2	
繰り返し2-3	f1 {f2}	2.4	{f1, f2}	f0, f1, f2	
繰り返し2-4	f3 {f1, f2, f3}	2.6	{f1, f2}	f0, f1, f2, f3	
繰り返し3-1	f0 {f0, f1, f2}	3.5	{f1, f2}	f0	
繰り返し3-2	f2 {f1}	2.2	{f1, f2}	f0, f2	
繰り返し3-3	f1 {f2}	2.4	{f1, f2}	f0, f1, f2	
繰り返し3-4	f3 {f1, f2, f3}	2.6	{f1, f2}	f0, f1, f2, f3	

【図 6】



【図 7】



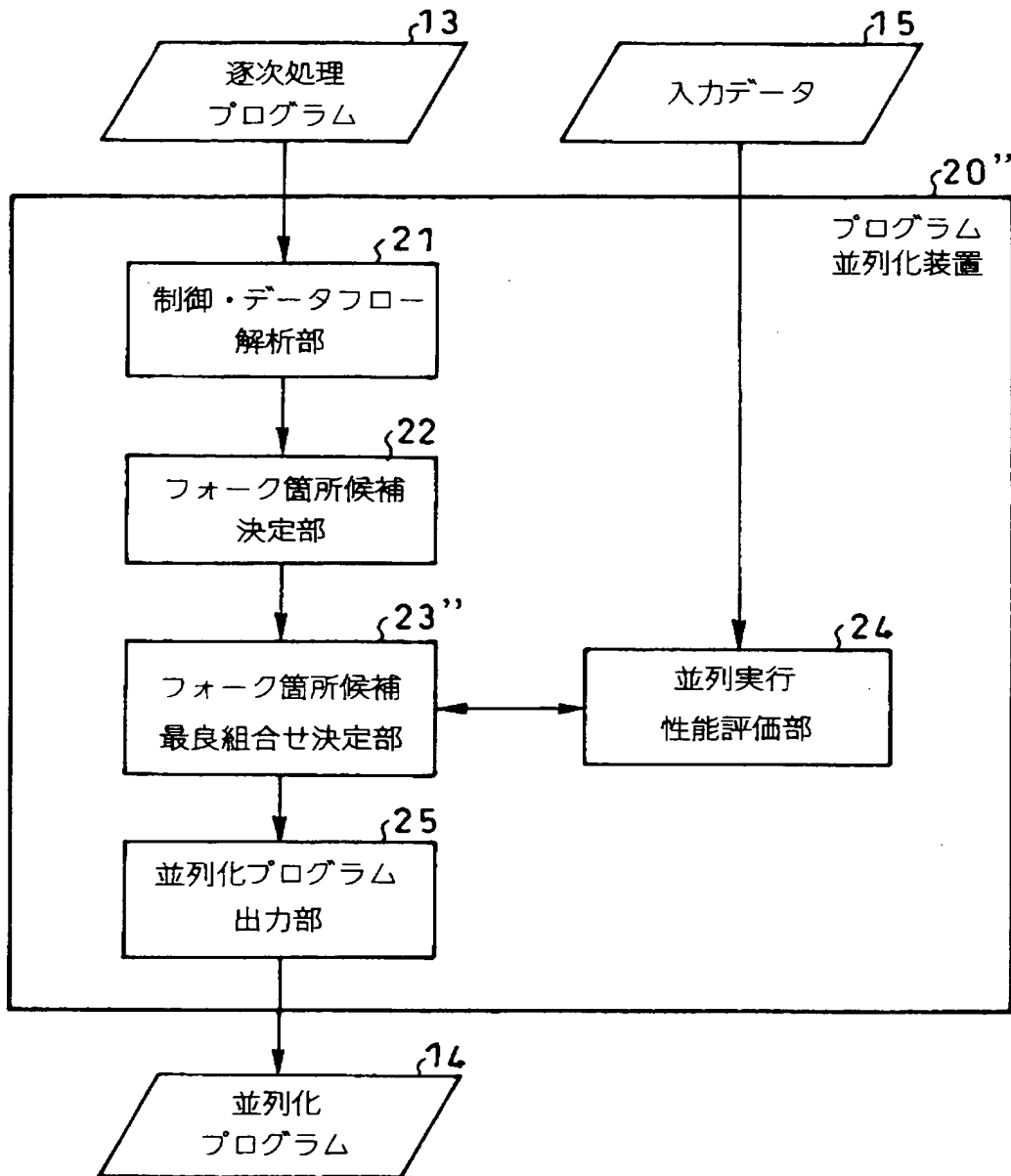
【図 8】

初期組合せ: { f0, f2 } (並列実行性能: 3.2倍)

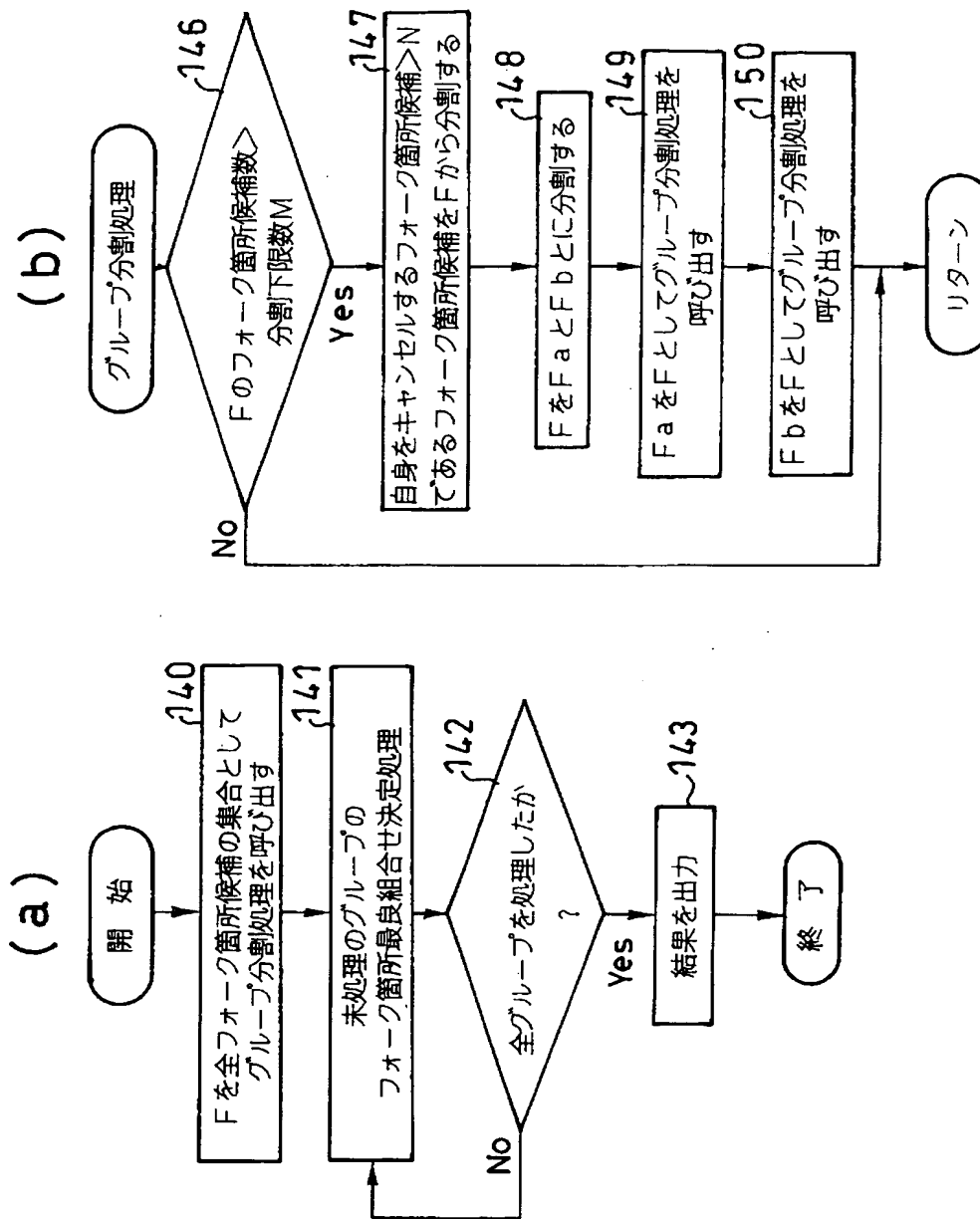
並列実行性能			
選択フオーク箇所候補		並列実行性能	
試験組合せ		最良組合せ	選択済フオーク箇所候補
繰り返し 1-1	f0 {f2}	2.4 {f0, f2}	f0
繰り返し 1-2	f2 {f0}	2.5 {f0, f2}	f0, f2
繰り返し 1-3	f1 {f1, f2}	3.8 {f1, f2}	f0, f1, f2
繰り返し 1-4	f3 {f1, f3}	2.8 {f1, f2}	f0, f1, f2, f3
繰り返し 2-1	f0 {f0, f1, f2}	3.5 {f1, f2}	f0
繰り返し 2-2	f2 {f1}	2.2 {f1, f2}	f0, f2
繰り返し 2-3	f1 {f2}	2.4 {f1, f2}	f0, f1, f2
繰り返し 2-4	f3 {f1, f3}	2.8 {f1, f2}	f0, f1, f2, f3



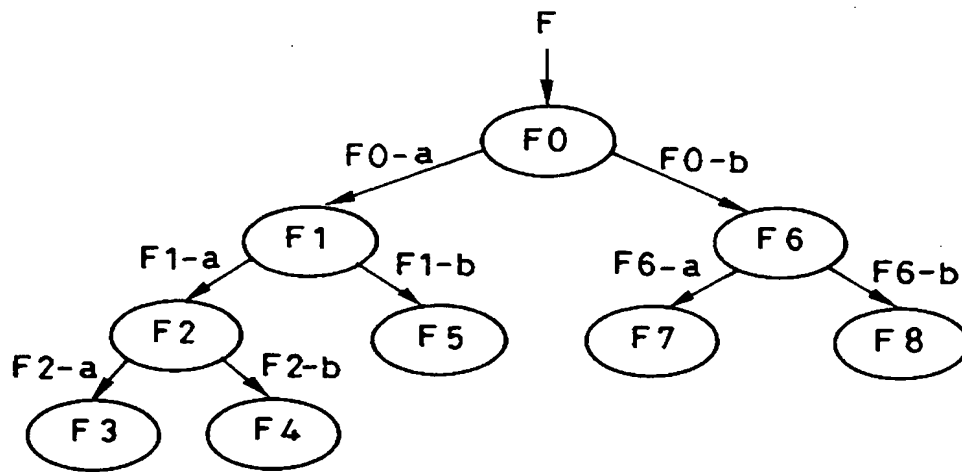
【図 9】



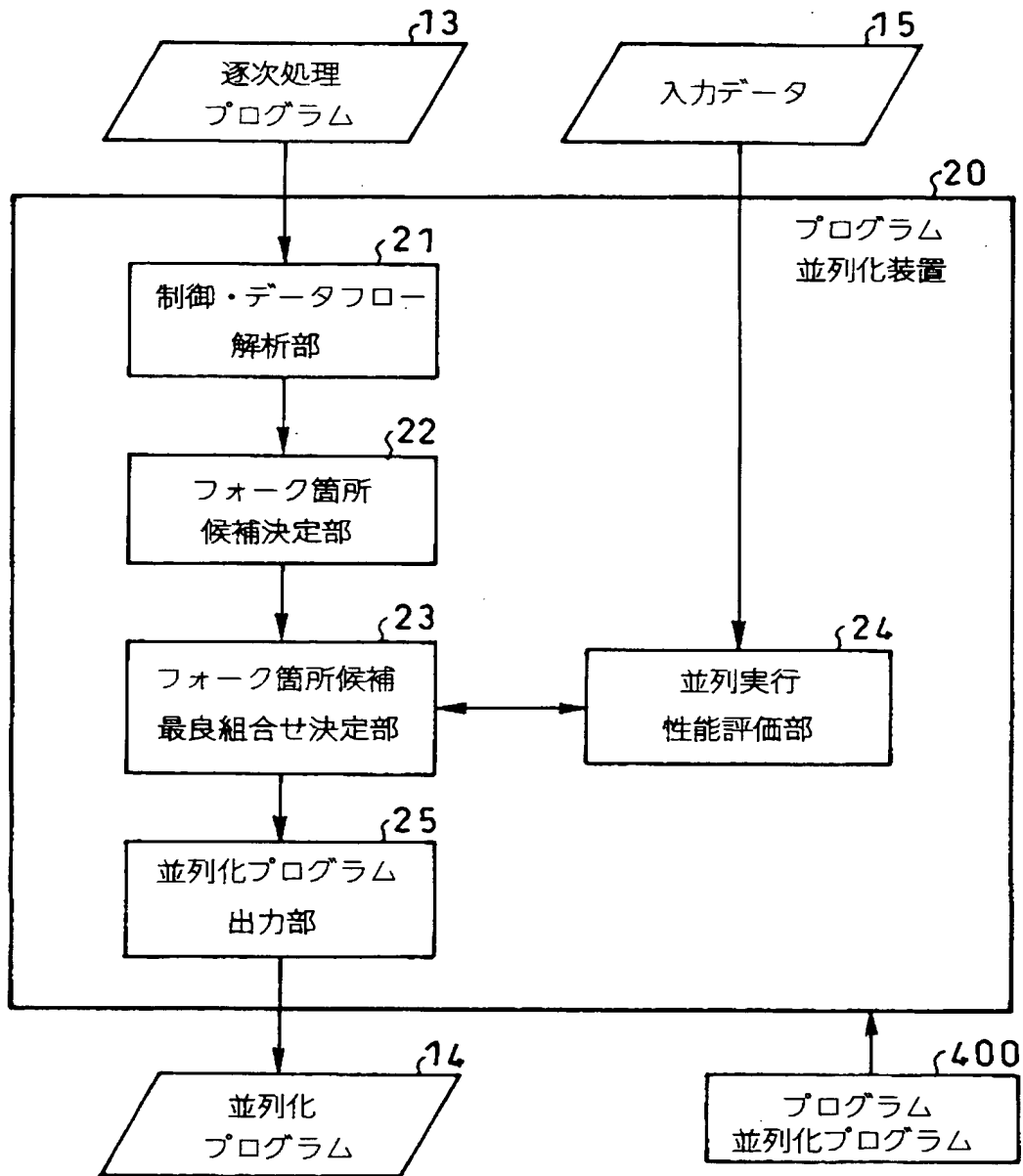
【図 10】



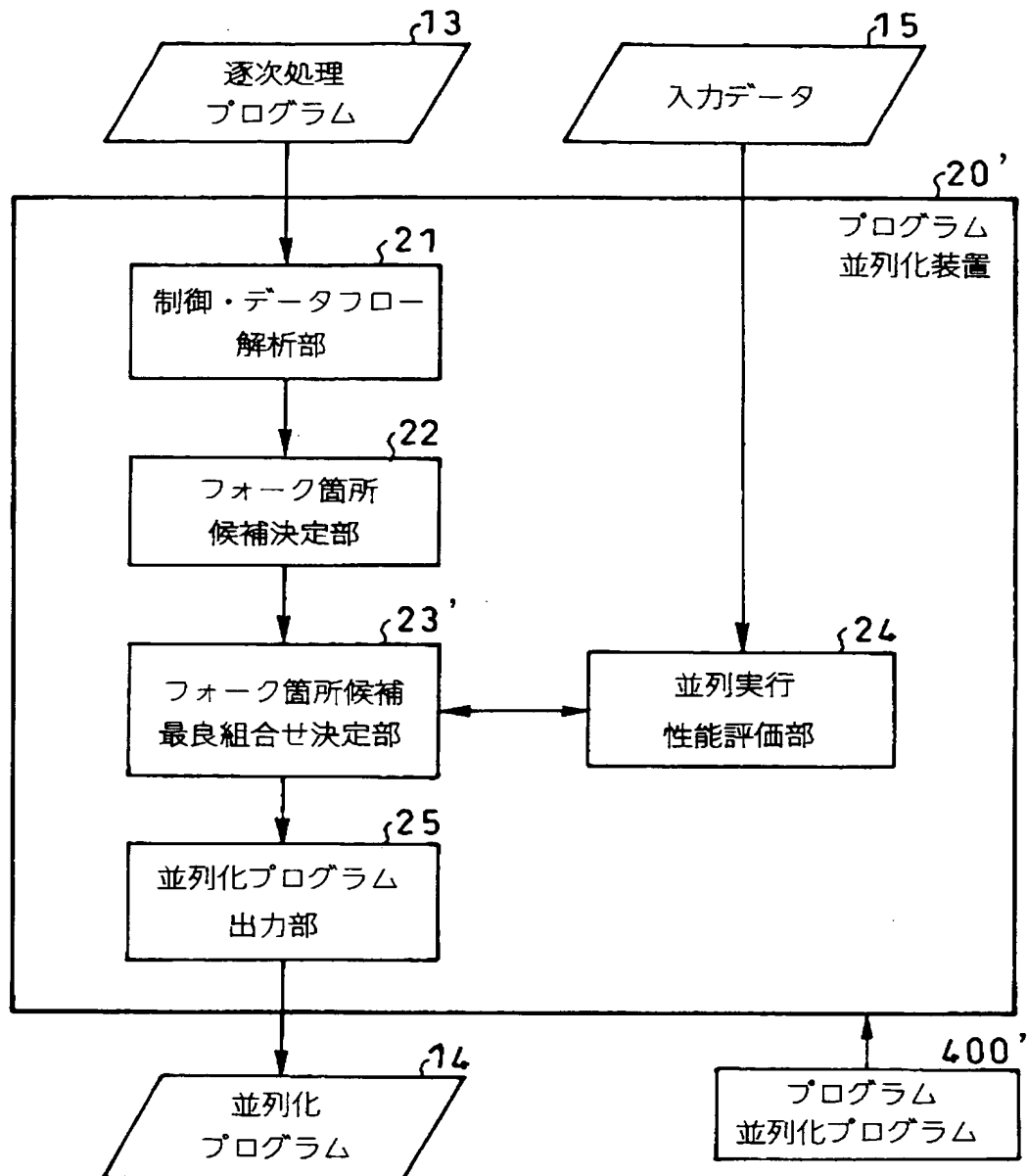
【図 11】



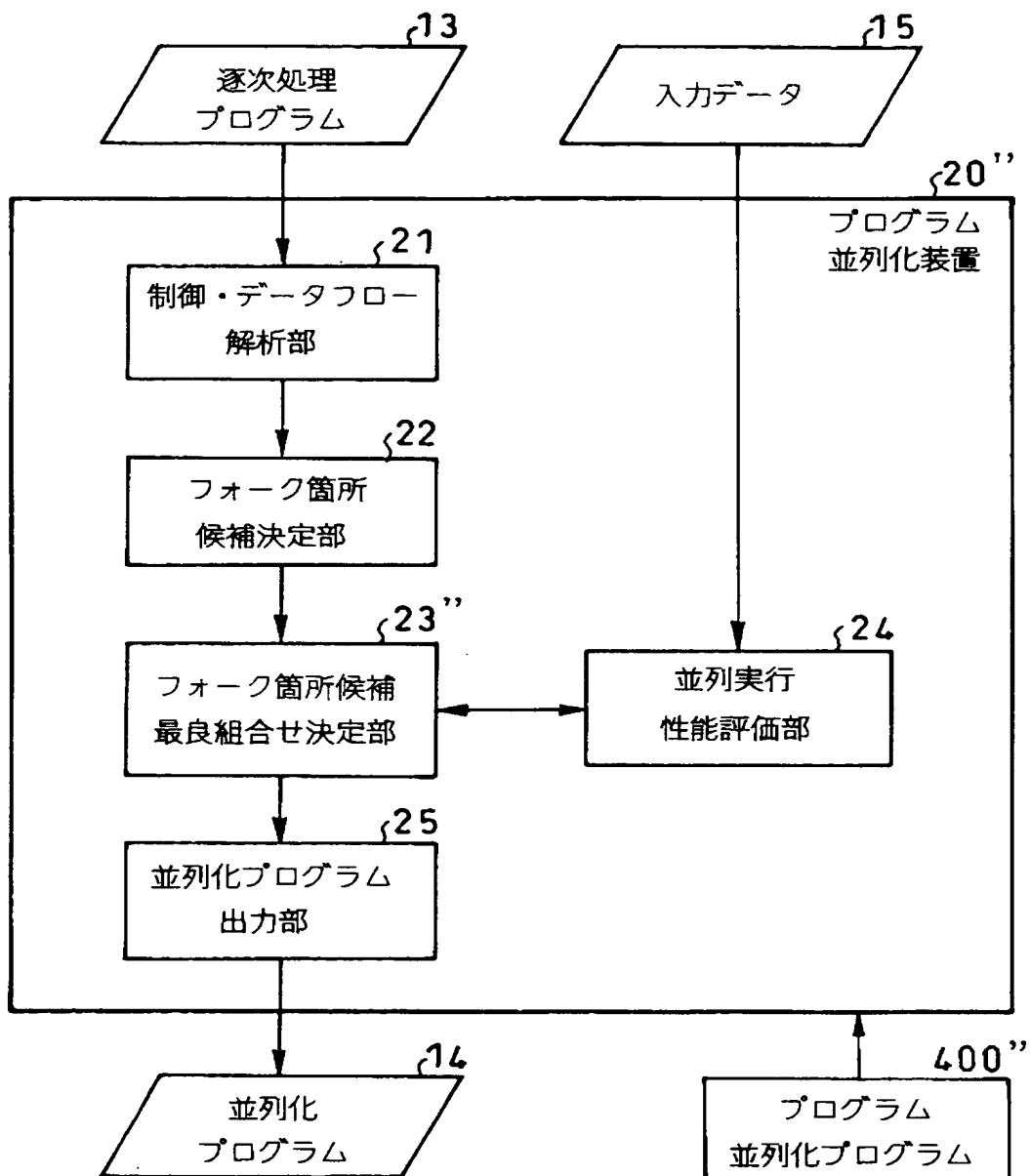
【図 12】



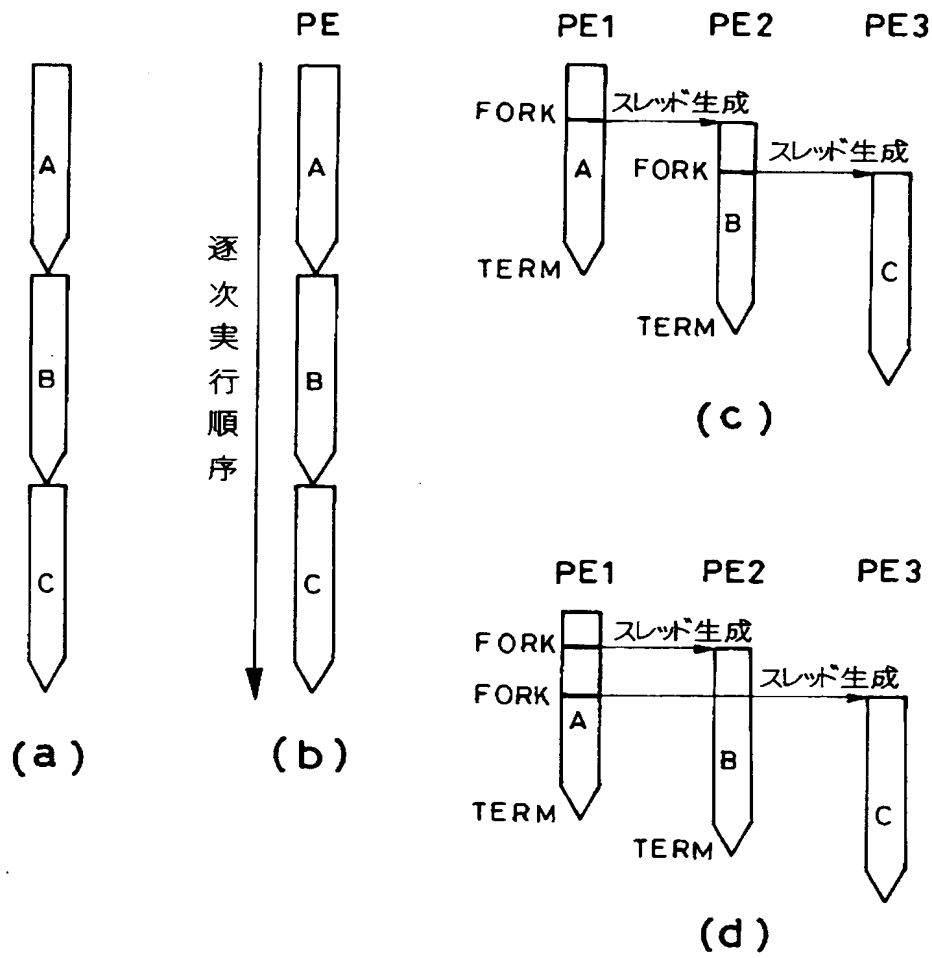
【図 13】



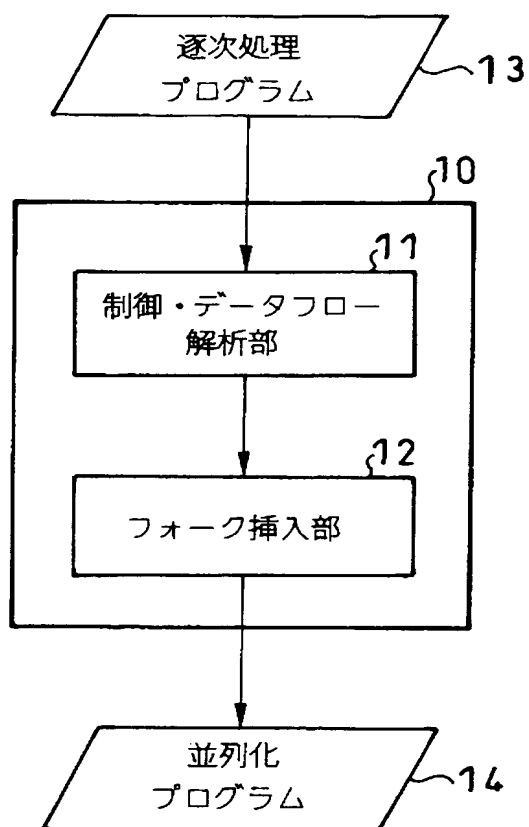
【図 14】



【図 15】



【図 16】





【書類名】 要約書

【要約】

【課題】 逐次処理プログラムを複数のスレッドに分割し複数のプロセッサで並列に実行するマルチスレッド実行方法向けに、並列実行性能のより高い並列化プログラムを高速に生成する。

【解決手段】 制御・データフロー解析部 2 1 は、逐次処理プログラム 1 3 の制御フローおよびデータフローの解析を行い、それを参照してフォーク箇所候補決定部 2 2 がフォーク箇所候補を決定する。フォーク箇所候補最良組合せ決定部 2 3 は、並列実行性能評価部 2 4 によってフォーク箇所候補の試行組合せの並列実行性能を評価した結果を基準にフォーク箇所候補の最良組合せを決定し、並列化プログラム出力部 2 5 はフォーク箇所候補の最良組合せに基づいてフォーク命令を挿入して並列化プログラム 1 4 を生成し出力する。

【選択図】 図 1

## 認定・付加情報

特許出願の番号	特願 2 0 0 3 - 0 9 3 0 7 6
受付番号	5 0 3 0 0 5 2 2 9 4 5
書類名	特許願
担当官	第七担当上席 0 0 9 6
作成日	平成 1 5 年 4 月 1 日

### < 認定情報・付加情報 >

【提出日】 平成15年 3月31日

次頁無

特願 2 0 0 3 - 0 9 3 0 7 6

出 願 人 履 歷 情 報

識別番号

[ 0 0 0 0 0 4 2 3 7 ]

1. 変更年月日

1 9 9 0 年 8 月 2 9 日

[変更理由]

新規登録

住 所

東京都港区芝五丁目 7 番 1 号

氏 名

日本電気株式会社